FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

IP Based Live Audio Selector

José Carlos da Rocha Lima

Mestrado Integrado em Engenharia Informática e Computação

Orientadora: Maria Teresa Magalhães da Silva Pinto de Andrade

25 de Julho de 2018

IP Based Live Audio Selector

José Carlos da Rocha Lima

Mestrado Integrado em Engenharia Informática e Computação

25 de Julho de 2018

Resumo

Devido à rápida e constante evolução tecnológica que se tem verificado nos últimos tempos, as tecnologias *IP* têm sofrido grandes alterações melhorando, consideravelmente a sua performance e fiabilidade.

Esta evolução tornou a tecnologia *IP* bastante atrativa para diversas indústrias que têm vindo cada vez mais a adotá-la para a transmissão de media, passando a realizar através de *software* operações que anteriormente eram implementadas por *hardware*.

O processo de virtualização da media para a camada de *IP* levanta grandes desafios, principalmente na distribuição de conteúdos audiovisuais de qualidade, em particular nos ambientes profissionais de produção audiovisual. Os débitos gerados são elevados e os requisitos são exigentes, nomeadamente em termos de atrasos na distribuição dos conteúdos, perdas de informação e sincronização entre as diferentes componentes media.

Esta dissertação pretende resolver alguns destes desafios , propondo uma abordagem para a comutação de diversos canais de áudio num ambiente de transmissão ao vivo, investigando quais as mais recentes normas e protocolos para o transporte de áudio na camada de *IP* utilizados pela indústria televisiva. O resultado final é um prototipo de uma aplicação capaz de transmitir diversos canais de áudio, permitindo a comutação entre eles. É também proposta uma abordagem para a sincronização dos diversos fluxos de dados media que existem num ambiente de produção audiovisual.

Abstract

Due to the fast and constant technological evolution that has taken place in recent years, IP technologies had suffered great changes considerably improving their performance and reliability.

This advance made the technology quite attractive for several industries that had been vowed to adopt it for the media transmission, starting to execute in software some operations that were previously implemented by hardware.

The media virtualization process for the IP layer represents major challenges, mainly in the distribution of quality audio-visual contents, particularly in professional audio-visual production environments. The debits generated are high and the requirements are demanding, namely in terms of delays in the distribution of content, loss of information and synchronization between the different media components.

This dissertation aims to solve some of these challenges and proposes an approach for the switching of multiple audio channels in a live broadcast environment, investigating the latest standards and protocols for transporting audio at the IP layer used by the broadcast industry. The end result is a prototype of an application capable of transmitting multiple audio channels and allowing switching between them. An approach is also proposed to synchronize the various media streams that exist in an audio-visual production environment.

Agradecimentos

Em primeiro lugar, dirijo um agradecimento especial aos meus pais e à minha irmã, pelo apoio incondicional, incentivo, paciência, amizade sempre demonstrados ao longo do meu percurso e também pelo esforço colocado para garantirem as condições necessárias à minha formação académica.

À minha orientadora, Professora Maria Teresa Andrade, por ter aceite este desafio e pela forma como orientou o meu trabalho. Estou grato pela cordialidade com que sempre me recebeu, pela utilidade das suas recomendações , mas, também, pela liberdade que me concedeu na realização deste trabalho.

À *MOG Technologies*, pela oportunidade e todas as condições necessárias para o desenvolvimento deste trabalho, em particular ao Engenheiro Alexandre Ulisses, pela proposta deste desafio.

Finalmente, gostaria de deixar um agradecimento especial aos Engenheiros Pedro Santos e Vasco Filipe que acompanharam de perto o desenvolvimento da dissertação, demonstrando sempre grande empenho, disponibilidade, paciência e interesse.

José Lima

..

"Everything is theoretically impossible, until it is done."

Robert A. Heinlein

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

IP	Internet Protocol
TCP	Transmission Control Protocol
AoIP	Audio Over IP
UDP	User Datagram Protcol
ADT	Abstract Data Type
SaaS	Software as a service
OTT	Software as a service
JT-NM	Join Task Force on Network Media
EBU	European Broadcasting Union
SMPTE	Society of Motion Pircture and Television Engnineers
VSF	Video Services Forum
CHIC	Cooperative Holistic View On Internet Content
IGMP	Internet Gourp Management Protocol
RTP	Real-time Transfer Protocol
MPEG	Moving Picture Experts Group
NTP	Network Time Protocol
UTC	Coordinate Universal Time
LAN	Local Area Network
PTP	Precision Time Protocol
SDI	Serial Digital Interface
NMOS	Network Media Open Specifications
AMWA	Advanced Media Workflow Association
AES	AudioEngineering Society
COTS	Commercial Off-The-Shelf
QoS	Quality of Service

Capítulo 1

Introdução

Os sinais de que a produção televisiva terá que expandir para o *IP* estão em toda a parte. Em termos de distribuição, com os avanços nas instalações da fibra ótica tem se vindo a verificar um aumento significativo na procura por serviços que distribuem conteúdo audiovisual através de fluxos de media pela Internet. Este tipo de serviços, aliado a uma boa conexão de Internet, tem vindo a modificar a forma como os consumidores acedem aos seus programas favoritos, quer estes sejam ao vivo ou pré-gravados[9].

De forma a não perderem consumidores, os distribuidores televisivos e as empresas relacionadas com media estão a sentir a necessidade de mudar o seu modelo de negócio, procurando soluções baseadas em *cloud* e tecnologias de virtualização, expandindo, assim, o seu público-alvo através de ofertas *SaaS (software as a service)*.

Deste modo, tem sido feito um grande investimento por parte da industria televisiva de forma a conseguir suportar estas novas formas no consumo de conteúdo audiovisual e, ao mesmo tempo, conseguir manter a compatibilidade com os métodos mais tradicionais utilizados atualmente[10].

1.1 Contexto / Enquadramento

Com a transição da indústria televisiva para infraestruturas que suportem o *IP*, de forma a agilizar e garantir que a existência de interoperabilidade, têm sido criadas normas para tornar possível a virtualização das produções televisivas. A *Join Task Force on Network Media (JT-NM)*, é um grupo formado pela *European Broadcasting Union (EBU)*, *Society of Motion Pircture and Television Engnineers (SMPTE)* e o *Video Services Forum (VSF)* que se juntou com o objetivo de definir estas normas.

Esta dissertação foi desenvolvida em ambiente empresarial na *MOG Technologies*, empresa envolvida no mercado de *broadcasting* desde 2007, fornecendo essencialmente soluções para ambientes de pós-produção.

Esta dissertação enquadra-se numa solução que a *MOG Technologies* está a desenvolver com vários parceiros, denominada de *Cooperative Holistic View On Internet Content*(*CHIC*)[11]. O trabalho desenvolvido neste dissertação será no futuro um contributo para um sub-projeto desta solução.

Este sub-projeto visa desenvolver um sistema de ingestão de conteúdos de televisão de alta resolução em tempo real. Devem ser utilizadas as normas mais recentes para a implementação de sistemas de contribuição utilizando *IP*, que visem permitir a utilização da Internet a grandes distâncias na ligação entre os sistemas de captação e a régie dos estúdios de realização. Utilizar-se-ão as tecnologias mais avançadas no desenvolvimento da solução, com o objetivo de resolver, sem perdas, os problemas associados à grande quantidade de dados a transmitir, requisito fundamental da atividade profissional.

O *workflow* proposto para esta solução é o representado na figura 1.1, sendo que o trabalho desenvolvido nesta dissertação só se irá focar na parte do transporte do áudio e da sincronização dos diferentes elementos.

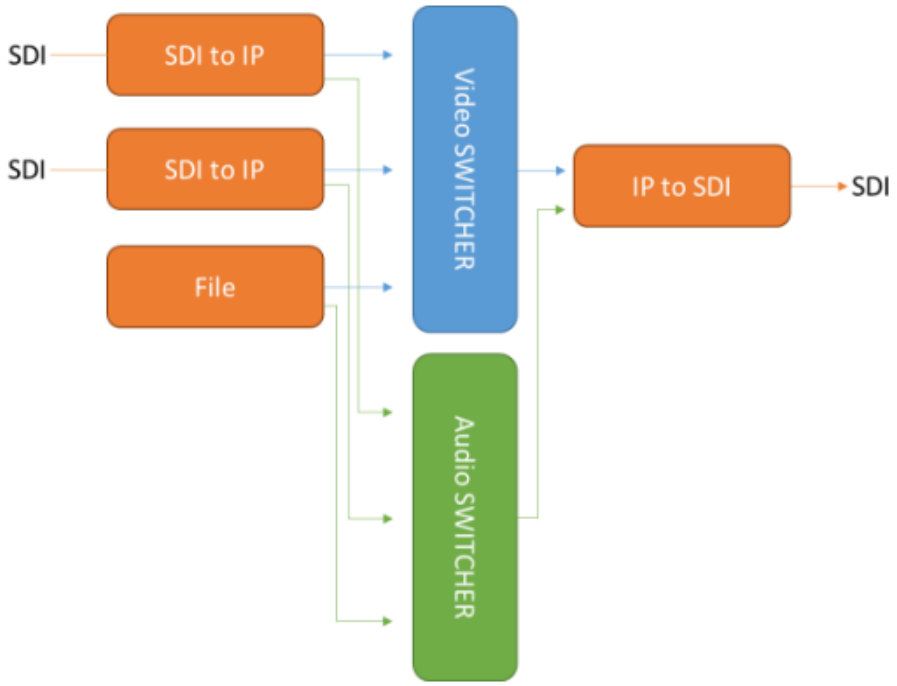

Figura 1.1: *Workflow* atividade projeto do *CHIC*

1.2 Motivação e Objetivos

Ao suportar o *IP*, os distribuidores de televisão serão capazes de distribuir conteúdo audiovisual personalizado para cada utilizador.

A transição de processos que antigamente eram realizados por *hardware* para aplicações sobre *IP* resultará numa baixa significativa dos custos da produção (poder de processamento, armazenamento) e , ao mesmo tempo, viabilizar uma maior flexibilidade, permitindo que escalem se necessário sem que sejam necessárias grandes alterações na sua arquitetura.

O objetivo principal desta dissertação é de desenvolver o protótipo de uma aplicação sobre *IP*, que seja capaz de ter vários canais de áudio como entrada e que permita a comutação entre estes canais de forma a que apenas um seja selecionado para a saída da aplicação protótipo.

Esta aplicação deve permitir, o mais breve possível, realizar a comutação através de um *web browser*, mantendo a qualidade e o sincronismo de todos os canais.

1.3 Estrutura da Dissertação

Para além do capítulo 1, este documento possui mais 7 capítulos. No capítulo 2 estuda-se o modelo *TCP/IP*, compreendo as diferentes vicissitudes da transmissão de dados sobre a camada de *IP*.

No capítulo 3 analisam-se os métodos de sincronização de máquinas numa rede que têm maior relevância em sistemas que necessitam de alta precisão.

No capítulo 4 reveem-se os sistemas de produção televisiva digital atuais e quais os mecanismos que foram criados para suportar a transição para estúdios *IP*.

O capítulo 5 estuda de que forma um sinal analógico de áudio é transformado num sinal digital e como é que as diferentes soluções existentes no mercado para o transporte deste sinal digital numa rede podem interoperar umas com as outras.

No capítulo 6 é proposta uma arquitetura para a virtualização de um misturador de áudio, com base na qual se apresenta o protótipo desenvolvido.

O capítulo 7 apresenta os testes que realizados para validar o protótipo desenvolvido, assim como análise dos resultados obtidos com esses testes realizados.

A dissertação termina com o capítulo 8, onde é feita uma avaliação dos objetivos propostos, bem como daquilo que se perspetiva para trabalho futuro.

Capítulo 2

Transporte de dados sobre *IP*

O modelo *TCP/IP* é um protocolo de comunicação utilizado para interligar dispositivos na internet. Este protocolo está dividido em quatro camadas: *Application*, *Transport*, *Internet* e *Network Interface*[12]. Cada uma destas camadas executa tarefas distintas de forma a garantir a integridade dos dados que circulam na rede.

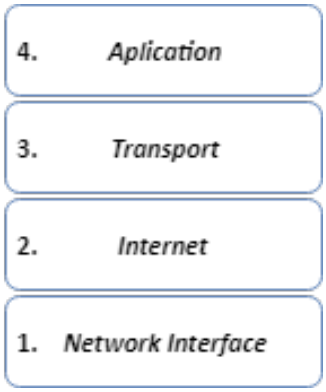

Figura 2.1: Modelo *TCP/IP*

Tendo em conta os objetivos desta dissertação é importante analisar as duas camadas superiores, nomeadamente application e transport. A camada *application* é uma abstração que engloba protocolos que realizam a comunicação entre aplicações e os protocolos de transporte *TCP/IP*, enquanto que a camada de *transport* é responsável por receber os dados da camada *application* e tratar do seu empacotamento.

2.1 Camada de Transporte

Numa rede, os dados são transportados através de três métodos:

- ***Unicast***
  Numa transmissão *unicast*, para cada mensagem enviada, existe apenas um recetor. É um método um-para-um (*one-to-one*).

- ***Broadcast***
  Numa transmissão *broadcast*, cada mensagem enviada pode ser recebida por qualquer nó da mesma rede sem exceção. É um método um-para-todos(*one-to-all*).

- ***Multicast***
  Numa transmissão *multicast*, cada mensagem enviada, é apenas recebida por nós subscritos a um determinado endereço *multicast*. É um método um-para-muitos(*one-to-many*).

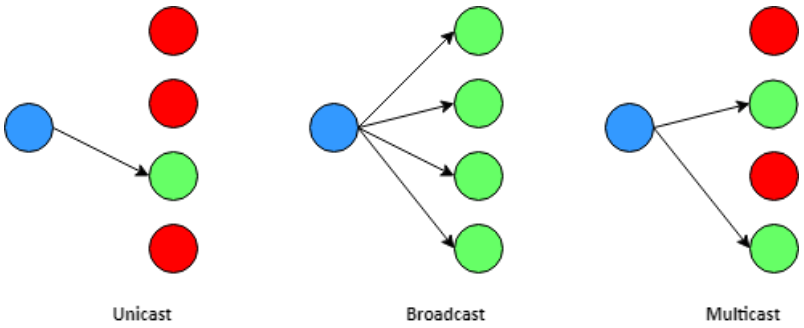

Figura 2.2: *Unicast, Broadcast*/ e *Multicast*

De forma a cumprir as suas funções, a camada de transporte utiliza dois protocolos, *UDP* e *TCP*.

- ***TCP****(transmission control protocol)*:
  É criado um canal seguro de comunicação virtual bidirecional entre dois dispositivos. Sempre que um pacote é recebido, o recetor envia uma mensagem ao emissor com a confirmação da receção do pacote, em caso de falha, os pacotes são transmitidos novamente, garantindo assim que os dados são recebidos na totalidade pelo recetor e pela ordem correta. A comunicação no *TCP* é feita unicamente por *unicast*.

- ***UDP****(user datagram protocol)*:
  Comparativamente ao *TCP* é um protocolo muito simples que fornece uma camada de transporte para dados, servindo apenas de empacotador para as aplicações acederem ao *IP*. Como não é estabelecida uma ligação previamente, o *UDP* não garante a ordenação e a entrega de pacotes. As comunicações podem ser *multicast* ou *broadcast*.

Apesar de o *TCP* ser um protocolo mais completo, em certos casos, é posta em causa a viabilidade da aplicação, devido à velocidade e sobrecarga introduzida pelo protocolo de ligação, sendo mais vantajoso utilizar o *UDP*. Em ambientes controlados, tais como redes privadas, as desvantagens do *UDP* são significativamente reduzidas.

2.2 IGMP

O *Internet Gourp Management Protocol(IGMP)*[13] é um protocolo de comunicação que permite a um nó anunciar na rede a que grupos de *multicast* está subscrito. o *IGMP* é um protocolo standard usado pelo *TCP/IP*. Este protocolo é muitas vezes utilizado em aplicações de transmissões de vídeo, permitindo uma maior eficiência no uso de recursos de forma a suportar estas aplicações.

2.3 *Real-time Transfer Protcol*

O *RTP*[1] é um protocolo de transporte de áudio e vídeo sobre redes *IP*. Cada pacote *RTP* contem um cabeçalho com informação relativa ao *sequence number*, *timestamp* e versão do protocolo, entre outros. A existência de um parâmetro customizável permite que o protocolo seja expandido.

Ao marcar cada pacote com um número sequencial, é possível reordenar os pacotes após a sua transmissão, independentemente da ordem em que foram recebidos.

Ao usar o *RTP* em conjunto com o *UDP* é possível maximizar a velocidade na transferência de informação e ao mesmo tempo manter a ordem dos pacotes.

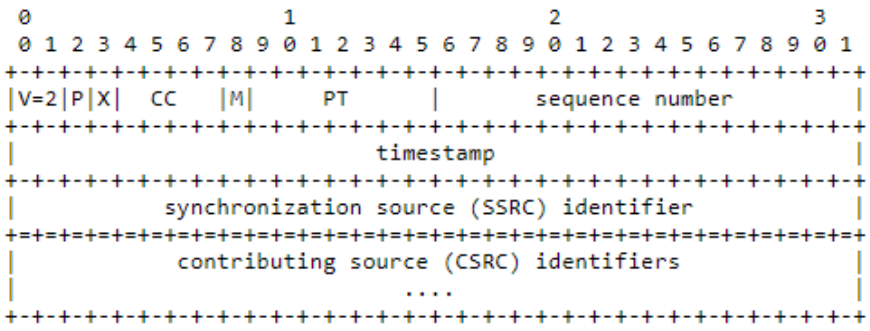

Figura 2.3: Pacote *RTP*[1]

2.4 *MPEG Transport Stream*

O *MPEG-TS* é um *container* utilizado para transportar e armazenar *media*. As *streams* de *MPEG-TS* sao compostas por um ou mais programas, descritos numa tabela chamada *Program Association Table (PAT)*.

Um programa é composto pela combinação de uma ou mais *streams* de *PES (Packetized Elementary Stream)*. Por exemplo, um programa pode conter uma vídeo *PES*, uma áudio *PES* e uma PES para legendas. O *Program Map Table (PMT)* armazena a informação de todos os programas presentes na *stream MPEG-TS*, como é possível observar na figura 2.4.

Uma *stream MPEG-TS* é composta por um grupo de pacotes *TS*, cada um tem um tamanho fixo de 188 *bytes*, 4 para o cabeçalho e 184 para o conteúdo.

As *streams MPEG-TS* devem de ser encapsuladas em pacotes *RTP* de forma a serem transportadas usando o *IP*. Cada pacote de *IP* contem um cabeçalho de *IP*, um cabeçalho *UDP*, um cabeçalho *RTP* e o número de pacotes *TS* que contem.

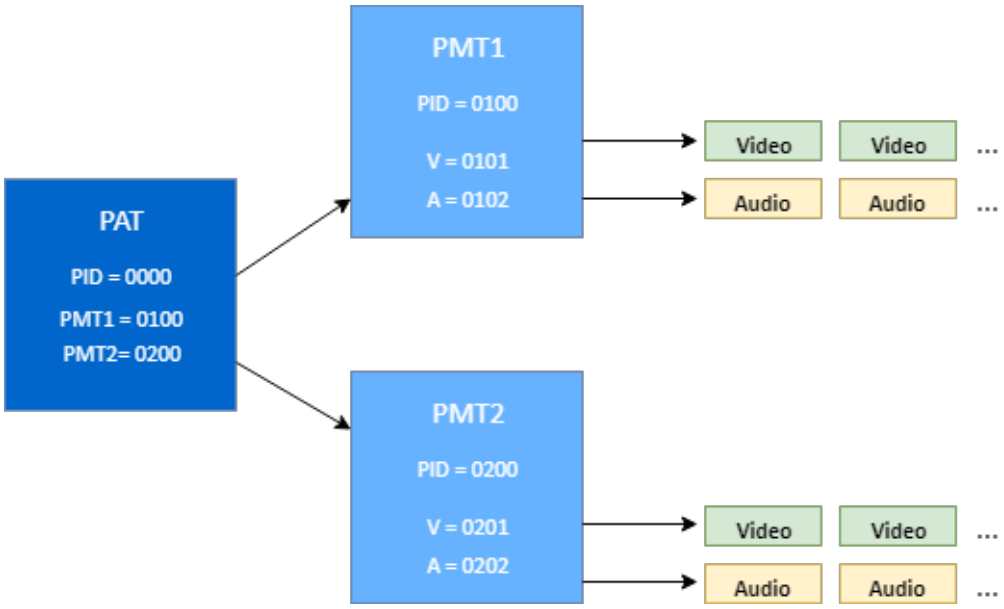

Figura 2.4: *MPEG PAT*[2]

Capítulo 3

Sincronização

Num sistema centralizado, o tempo não é ambíguo. Quando um processo precisa de conhecer o tempo, é feita uma chamada ao sistema e o *kernel* dá-nos essa informação. Se por exemplo, um processo ***A*** pedir o tempo do sistema e posteriormente um processo ***B*** fizer o mesmo pedido, o valor que ***B*** recebe vai ser maior (ou possivelmente igual) ao que ***A*** recebe. Num sistema distribuído, conseguir esta conformidade temporal não é tão trivial.

A sincronização de relógios é um requisito obrigatório para qualquer sistema distribuído. Quando é preciso saber a que altura ocorreu determinando evento, o tempo muita é, frequentemente, a única referência entre os diversos dispositivos na rede e, por isso, é necessário que a distribuição do tempo seja fidedigna e de alta precisão, de forma a que todos os nós na rede sincronizem os seus relógios a uma determinada fonte.

Tendo em conta o objetivo desta dissertação é importante explorar dois protocolos de sincronização de relógios numa rede. Sendo estes o *Network Time Protocol (NTP)*[14] e o *Precision Time Protocol (PTP)*[4].

3.1 *Network Time Protocol*

O *NTP*[14] é um dos protocolos mais antigos de sincronização de relógios numa rede. O cliente *NTP* inicia uma troca de mensagens *time-request*, como resultado desta troca, o cliente é capaz de calcular o *delay* da ligação ao servidor e o *offset* do relógio local. Depois de obter estes valores é possível ajustar o seu relógio local de forma a sincronizar com o relógio do servidor. Depois de sincronizar, o cliente atualiza o seu relógio a cada 10 minutos, sendo necessário apenas uma troca de mensagens. Esta troca é feita por *UDP*.

Os servidores de *NTP* espalhados pelo mundo, têm acesso a relógios atómicos de alta precisão e a relógios *GPS*. O *NTP* utiliza o *Coordinate Universal Time (UTC)* para sincronizar os relógios de máquinas com alguma precisão, numa *local area networks (LAN)* na ordem do milissegundo e dezenas de milissegundo sobre a *internet*.

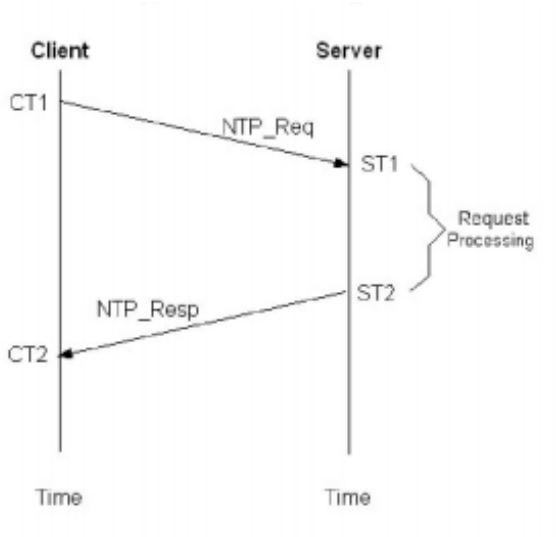

Figura 3.1: Mecanismo de *time-request* do *NTP*[3]

As camadas de separação até à fonte de *UTC* são chamadas de *strata*. Um relógio de referência, ou seja, que recebe o tempo diretamente de um transmissor dedicado ou de um sistema de navegação por satélite é categorizado por *stratum-0*. Um computador diretamente ligado a um relógio de referência é *stratum-1*, um computador que recebe o tempo do *stratum-1* pertence ao *stratum-2* e assim sucessivamente. A precisão vai reduzindo ao longo das camadas.

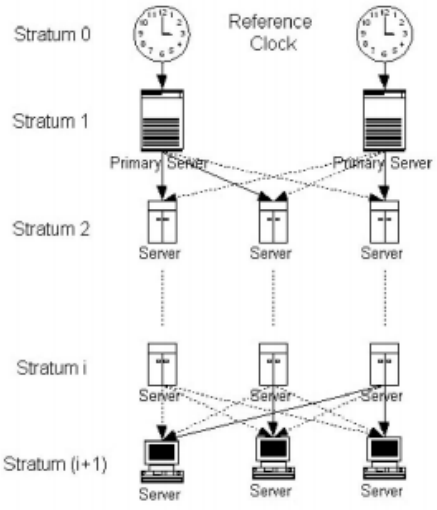

Figura 3.2: Hierarquia típica de *NTP*[3]

3.2 *Precision Time Protocol*

IEEE 1588[4] é um standard para sincronização de alta precisão entre máquinas numa rede controlada (IEEE 1588-2008 “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems”). Este standard foi introduzido inicialmente para ser utilizado na indústria da automação com o objetivo de obter uma sincronização precisa abaixo do microsegundo. Ao ganhar popularidade começou a ser utilizado por outros grupos, como por exemplo: telecomunicações, distribuição de energia elétrica e aplicações militares.
Ao utilizar uma arquitetura de slave-master, o *PTP* é capaz de distribuir um relógio de referência de uma dada fonte (*Grandmaster*) para um ou vários destinatários (*Slaves*) na mesma rede. No geral, a representação de um sistema básico de *PTP* pode ser vista na figura 3.3.

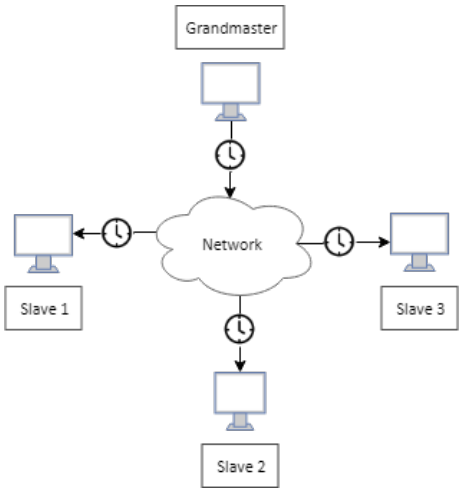

Figura 3.3: Exemplo arquitetura *Slave-Master PTP*

A sincronização entre o *Grandmaster* e os seus *Slaves* no *PTP* é conseguida pela troca de um conjunto de mensagens através da rede, como podemos ver na figura 3.4.

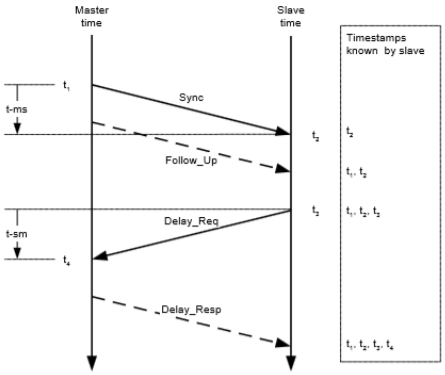

Figura 3.4: Mecanismo básico de sincronização[4]

Uma vez que todas as mensagens são enviadas via *multicast*, o *Grandmaster* começa o processo de sincronização com os seus *Slaves* ao completar a seguinte sequência de eventos:

1. O *Grandmaster* envia uma mensagem de ***Sync*** para os seus *Slaves* e estes registam a *timestamp* de receção **(t2)**.
2. Os *Slaves* recebem uma mensagem ***Follow_up*** do *Grandmaster* com a *timestamp* a que a mensagem ***Sync*** foi enviada **(t1)**.
3. Cada *Slave* envia uma mensagem de ***Delay_request*** ao *Grandmaster* e regista a *timestamp* de envio **(t3)**.
4. O *Grandmaster* regista a *timestamp* a que recebe a mensagem ***Delay_request*** **(t4)**.
5. Os *Slaves* recebem uma mensagem ***Delay_response*** do *Grandmaster* com a *timestamp* **t4**.
6. Os *Slaves* com as 4 *timestamps* conseguem calcular o *offset* do seu relógio em relação ao do *Grandmaster* e atualizam o seu relógio.

O *Offset* do *Slave* em relação ao *Grandmaster* é calculado da seguinte forma (3.1):

$$\begin{aligned} &< \textbf{offsetFromMaster} >=< \textbf{syncEventIngressTimestamp} > \\ &- < \textbf{preciseOriginTimestamp} > - < \textbf{meanPathDelay} > \end{aligned} \tag{3.1}$$

Substituindo na equação (3.1) as variáveis pelas *timestamps* obtidas na troca de mensagens, temos a seguinte equação(3.2):

$$< \textbf{offsetFromMaster} >= \textbf{t1} - \textbf{t2} - < \textbf{meanPathDelay} > \tag{3.2}$$

O *delay* que existe na transmissão de mensagens entre o *Grandmaster* e os *Slaves* na rede é calculado pelo mecanismo *delay request-response*. Este mecanismo mede o valor do *meanPathDelay* da seguinte forma(3.3):

$$< \textbf{meanPathDelay} >= \frac{(\textbf{t2} - \textbf{t3}) + (\textbf{t4} - \textbf{t1})}{\textbf{2}} \tag{3.3}$$

3.2.1 Requisitos para maior precisão na sincronização

De forma a garantir que existe uma precisão na sincronização abaixo dos 100 nanosegundos, é preciso garantir que os pacotes *PTP* tenham uma prioridade na rede maior que os restantes pacotes, de forma a serem entregues o mais rápido possível a todos os nós.

A geração de *timestamps* por parte de todos o nós também deve de ser feita com o auxilio de *hardware* próprio.

Também é possível gerar *timestamps* através de *software*, mas isso trás uma penalização na precisão obtida (tipicamente os resultados são entre os 10 a 100 microssegundos, no caso em que apenas o *slave* utiliza *software* na criação das suas *timestamps*).

Capítulo 4

Produção Televisiva Digital

4.1 *SDI*

Serial Digital Interface (SDI) é um conjunto de interfaces para vídeo digital normalizada pela *SMPTE (The Society of Motion Picture and Television)*. Desde os anos noventa que estas interfaces são um *standard* na industria televisiva.

O *SDI* apenas está disponível em equipamentos profissionais e possibilita o transporte de áudio e vídeo sem compressão em instalações televisivas. Sempre que surge um novo formato, é necessário adaptar as infraestruturas de *SDI* existentes para a que estas possam suportar os novos formatos[15].

4.1.1 Sistemas atuais de produção *SDI-BASED*

Um sistema típico de produção ao vivo *SDI-based* é constituído por várias ligações. Enquanto que o sinal de vídeo é transportado por cabos *SDI* ligados a um *router SDI*, que distribui a ligação para vários pontos, o sinal de áudio, muitas vezes, transportado numa rede separada, suportada por um *router* de áudio. Com o aparecimento dos chamados *hybrid routers*, as redes *audio-video* estão a tornar-se cada vez mais integradas. Para além das redes *audio-video*, existe também uma rede de *timing*, que transporta, os sinais responsáveis pela sincronização, para todos os equipamentos de produção, tais como câmaras, *switchers* de produção, e monitores. Existe também uma rede de controlo que transporta sinais responsáveis pela administração do sistema, monitorização e controlo. Estes sistemas de *SDI* podem ser então representados por três planos: *media*, *timing* e rede de controlo. A figura 4.1 representa um destes sistemas.

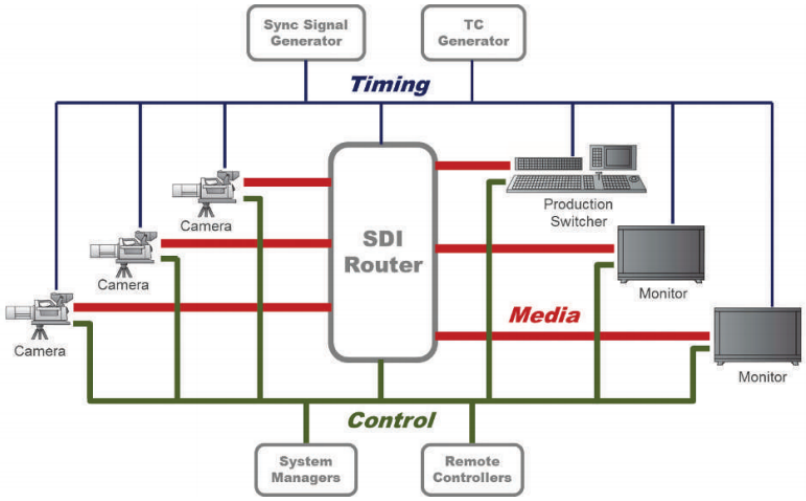

Figura 4.1: Sistema de produção *SDI*[5]

4.2 IP

Com o aumento da qualidade que se tem vindo a observar no surgimento de novos formatos, a utilização de *SDI* começa a ser limitativa devido à dimensão da infraestrutura necessária para suportar estes formatos.

A transição para IP começa, assim, a ser cada vez mais comum. Sendo uma solução *packet-based*, o *IP* é agnóstico a formatos. Neste caso é apenas *data* que está a ser transportada.

O facto de um cabo *IP* ser bidirecional, ao contrário do *SDI*, é um dos aspetos que torna uma infraestrutura *IP* bastante mais flexível e com custos mais baixos.

4.2.1 Sistemas atuais de produção *IP-BASED*

Nestes sistemas, a interface de ligação do equipamento de produção em vez de ser um *router SDI* passa a ser um conjunto de *IP switchers*, tornando a comunicação por *IP* em vez de *SDI*. Os equipamentos passam a estar todos ligados a *IP switchers* e estes fazem a distribuição dos sinais dos três planos definidos no sistema *SDI*. Para o plano de *media* todos os dispositivos estão interligados pelos *IP switchers*. Para o *timing*, é utilizado o *Precision Time Protocol (PTP)*[4] e um *Grandmaster* distribui tempo com precisão pela rede. Para o plano de controlo, são utilizados *softwares* específicos ligados aos equipamentos de produção por *IP*. A figura 4.2 represnta um sistema baseado em *IP*.

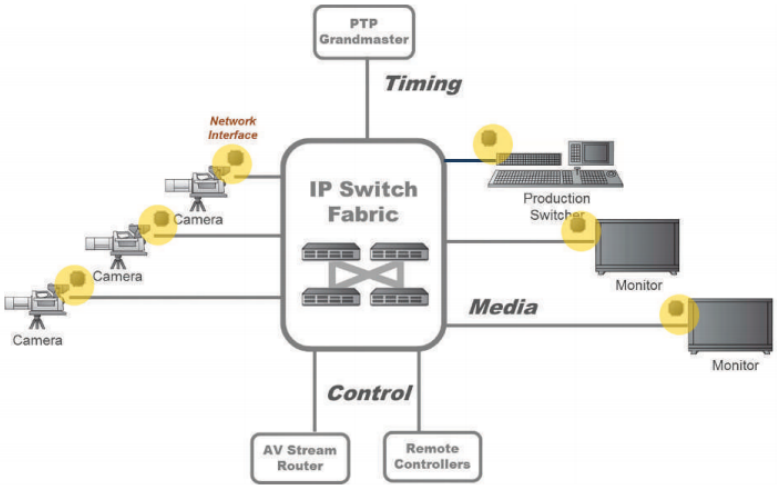

Figura 4.2: Sistema de produção *IP*[5]

4.3 Transição da industria para *IP*

4.3.1 *Joint Task Force on Network Media*

Com as vantagens que o *IP* tem face ao *SDI*, com custos mais reduzidos e maior flexibilidade da infraestrutura, cada vez mais se tem vindo a adotar este tipo de solução. Uma vez que já foi feito um grande investimento pela indústria a transição para o *IP* será gradual.

De forma a simplificar este processo, tem sido realizado trabalho na definição de novos *future-proof standards* que são capazes de ser implementados em cenários práticos. *The Joint Task Force on Network Media(JT-NM) reference architecture*[6] foi desenvolvida de forma a facilitar a transição dos equipamentos de *broadcasting SDI* para *IP*. Os trabalhos realizados nesta arquitetura de referência foram realizados pela *European Broadcasting Union*, *Society of Motion Picture* e *Video Services Forum*. O *JT-NM* não é um *standard* definitivo, mas fornece uma série de boas práticas, recomendações e *frameworks* que são úteis e capazes de serem implementadas em novos produtos comerciais ou cenários de negócio.

O foco desta nova arquitetura é facilitar a interoperabilidade de dispositivos e *software* de diferentes fabricantes na transição de *SDI* para *IP*.

O *core* numa produção de *media* em *IP* é a rede, tipicamente uma rede *Ethernet* baseada em pacotes. Os *Nodes* são ligados à rede de forma a provisionar infraestrutura adicional tal como poder de processamento e armazenamento. *Devices* podem ser serviços fornecidos por *software* ou dispositivos físicos, tais como microfones, são lançados em *Nodes*, nós existentes na rede, de forma a fornecerem as *Capabilities*, aptidões, necessárias para completar determinadas tarefas (*Tasks*).

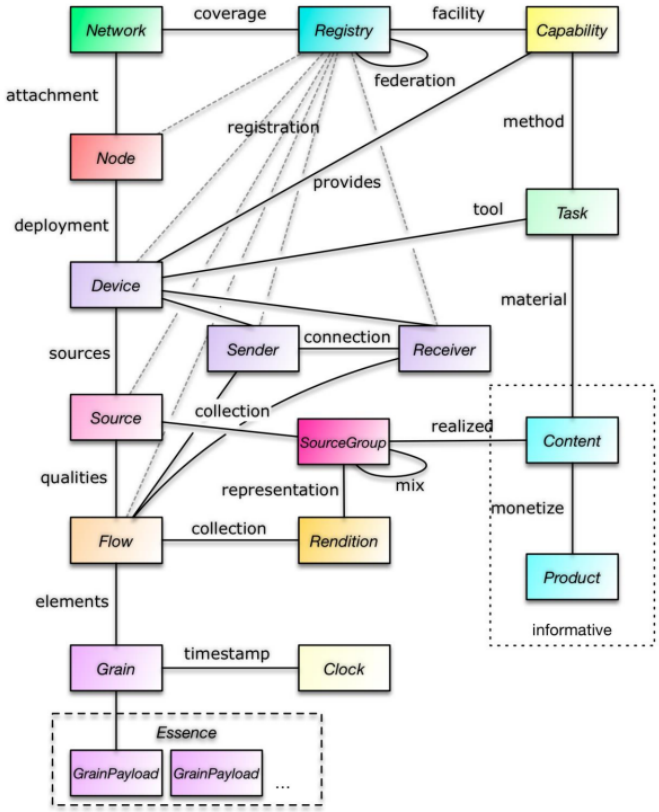

Figura 4.3: Modelo conceptual da arquitetura de referência *JT-NM*[6]

Dispositivos (*Devices*) podem ser fontes (*Sources*), origem de dados provenientes de um dispositivo, chamados de essências (*Essences*). Estas essências são movidas na rede como *Grain Payloads* que são subdivididos em pacote de rede para serem transportados.

Para que as essências sejam movidas na rede, é necessário que haja uma conexão entre o diapositivo que envia, transmissor (*Sender*), e o dispositivo que recebe, recetor (*Receiver*). A informação, incluindo as essências, é transferida entre transmissores e recetores na forma de fluxos (*Flows*). Cada fluxo é constituído por grãos (*Grains*), onde cada grão é composto por conteúdo *media*(vídeo, áudio, *metadata*) e uma *Timestamp*. A *Timestamp* representa o instante em que o grão foi criado e é gerada pelo relógio presente no nó (*Node*). Todos os nós devem de estar sincronizados, utilizando o *PTP*[4].

O *Registry* concede a conexão entre recetores e transmissores, fornecendo meios para que os fluxos, nós e dispositivos se registem de forma a conseguirem se encontrar na rede.

De maneira a que a informação possa ser devidamente articulada e utilizada na definição de novos *worflows*, o modelo estabelecido pelo *JT-NM* tem por base três blocos fundamentais: *Timing*, *Identity* e *Registration and Discovery*.

- ***Timing***

  De forma a garantir consistência em operações efetuadas e consequentemente garantir que

os *Flows* estão corretamente alinhados, cada *Grain* contem uma *timestamp*.

- ***Identity***
  Cada elemento presente na infraestrutura deve ser fácil e unicamente identificável, de forma a ser referenciado e utilizado.

- ***Registration and Discovery***
  Os *Nodes* presentes na rede devem de se registar a si próprios e aos *Devices*, *Sources*, *Flows*, *Senders* e *Receivers* que disponibiliza na rede, para que os outros *Nodes* os possam descobrir e obter informação que necessitam de cada um.

4.3.2 *NMOS*

Networked Media Open Specifications(NMOS)[7] são um conjunto de especificações que suportam a transição da indústria profissional de media audiovisual para um arquitetura "*fully-networked*". O *NMOS* é concebido pela *Advanced Media Workflow Association (AMWA)*.

As especificações do *MNOS* são desenvolvidas tendo a arquitetura de referência do *JT-NM* como base, mantendo a interoperabilidade desejada.

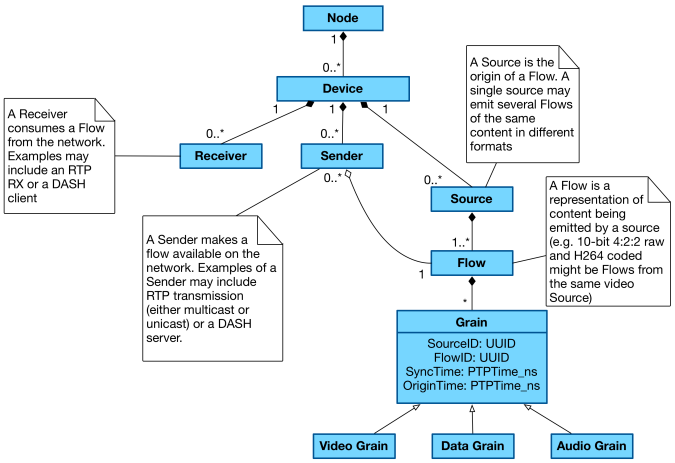

Figura 4.4: Modelo do *NMOS*[7]

No *NMOS* também são descritos dois mecanismos, *Registration* e *Discovery Specification*,

4.3.3 *SMPTE 2022*

O *SMPTE ST 2022* é um conjunto de normas publicadas pela *Society of Motion Picture and Television (SMPTE)* que permitem o uso da tecnologia IP na industria do *broadcasting*. O *SMPTE ST 2022-6* [16] define como deve ser transportado o sinal *SDI* sobre o *IP* usando o protocolo *RTP*.

Ao encapsular a carga de informação do *SDI* em pacotes de *IP*, esta norma permite que o sinal de *SDI* possa ser transportado numa rede *Ethernet* e, após transmissão, reconstruir o sinal *SDI*.

Apesar de garantir interoperabilidade entre vários dispositivos *SDI*, uma das grandes limitações do *ST 2022*, que o torna pouco flexível, é que não existe separação dos vários elementos do sinal *SDI*. Se, por exemplo, for necessário modificar o áudio que faz parte de um fluxo *SDI* que é transportado com a parte de vídeo correspondente, existe um aumento da sobrecarga do sistema quando é necessário alterar um elemento.

4.3.4 *SMPTE 2110*

Tal como o *2022*, o *ST 2110* define um conjunto de normas que utilizam a tecnologia de *IP* para o transporte de media. Ao contrário do *ST 2022*, que junta o vídeo, áudio e *metadata* num fluxo de *IP* único, o *ST 2110* divide cada sinal em fluxos independentes.

Utilizar o *ST 2110*, com a transmissão do vídeo, áudio e *metadata* em separado, garante um aumento na eficiência e permite que os recetores selecionem apenas os fluxos que pretendem para as suas operações.

Dentro da família do *ST 2110*, o *ST 2110-10* [17] introduz o *ST 2059 (PTP)* para distribuir uma base de tempo por todos os dispositivos presentes no sistema, de forma a conseguirem marcar cada pacote *RTP*, que depois é utilizado pelos recetores para conseguirem alinhar cada essência corretamente. O *ST 2110-30* [18] tem como foco o transporte de áudio *PCM*, especificamente o *AES67*[19].

O *JT-NM NMOS* é a peça responsável pela interoperabilidade entre dispositivos no standard, definindo como os dispositivos comunicam entre si na rede e como transportam a media.

Capítulo 5

Áudio sobre *IP*

5.1 Pulse Code Modualtion (PCM) Áudio

Pulse code modulation é um método utilizado para converter um sinal analógico num sinal digital, para que este sinal possa ser transmitido através de uma infraestrutura de rede. *PCM* é apresentado em forma binária, tendo apenas dois estados possíveis(0 e 1). É possível voltar a transformar o sinal digital em sinal analógico através do método de demodulação. O processo de modulação está dividido em três passos: *Sampling*, *Quantization* e *Encoding*.

- ***Sampling***
  Processo de medição e conversão da amplitude de um sinal continuo no tempo num sinal discreto. Um *Sample* é um valor ou um conjunto de valores num determinando instante. O *Sampling Rate* representa o número de *samples* por segundo que são utilizados de um sinal analógico para ser representado num sinal digital.

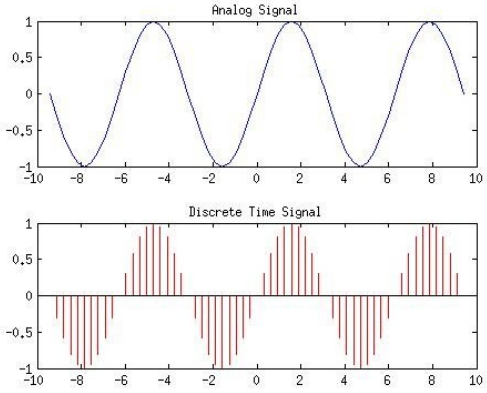

Figura 5.1: Sinal Analógico e sinal discreto

- ***Quantization***

  De forma a transmitir os valores das amostras obtidos no processo de *sampling* é necessário representar cada valor em forma numérica. Este processo requer uma quantização, onde o valor de cada amostra é arredondado para o valor numérico mais próximo num determinado conjunto numérico. É um processo complexo de arredondamento que envolve alguma imprecisão, o que torna a conversão de um sinal analógico para digital uma aproximação e não uma cópia exata.
  Quanto maior for o número de bits disponível para representar os valores arredondados, mais próximos estes vão estar do valor real da amostragem, reduzindo o erro da quantizaçao.

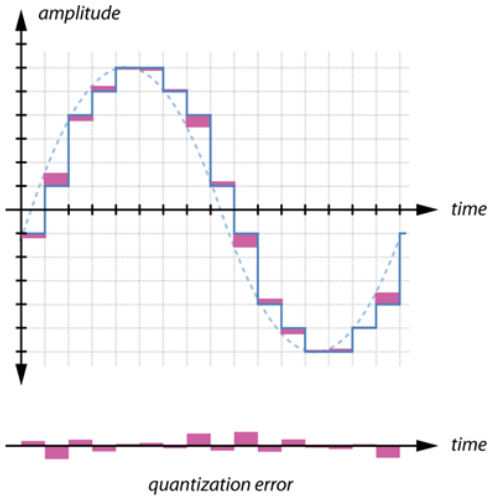

Figura 5.2: Quantização

- ***Encoding***

  No *encoding* cada valor aproximado na quantização é convertido para binário de forma a que o sinal possa ser guardado e transmitido em formato digital.

5.2 Soluções Áudio sobre *IP*

Com o aumento da procura de soluções de áudio sobre *IP*, têm surgido diversos sistemas de rede que utilizam o *IP* para permitir que o áudio seja distribuído por diversos dispositivos ligados a uma rede escalável. Sendo algumas destas soluções o *Ravenna*[20], *Dante*[21], *LiveWire*[22], *WheatNet*[23] e *Q-LAN*[24].

5.3 *AES67*

AES67 [19] é uma norma de interoperabilidade para áudio sobre *IP*, foi desenvolvido pela *Audio Engineering Society*, e tem como funcionalidade criar interoperabilidade na camada de transporte entre as várias soluções já existentes.

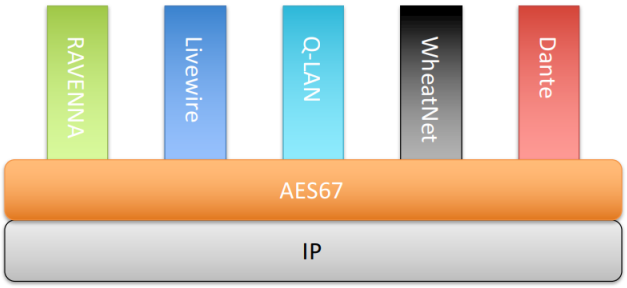

Figura 5.3: Interoperabilidade *AES67*[8]

Este standard foi concebido de forma a operar em infraestruturas de rede de troca de pacotes *COTS(Commercial Off-The-Shelf)*, isto é, se configuradas corretamente, a mesma rede pode ser partilhada com outro tráfego sem reduzir a performance na transmissão do áudio. O *AES67* baseia-se nos seguintes princípios fundamentais: Sincronização, Transporte, Codificação e Transmissão, por fim, Informação de sessão.

1. **Sincronização**
   A sincronização entre todos os transmissores e recetores do *AES67* é conseguida através da distribuição de tempo com precisão a todos os nós participantes. O *AES67* especifica que *standard IEEE1588-2008*(também conhecido como *Precision Time Protocol*) deve ser utilizado para a distribuição de tempo. Devem também ser gerados *Media Clocks*, que são relógios que têm uma relação fixa com o relógio distribuído na rede. Este *Media Clock* deve ter o mesmo *rate* que a frequência de *sampling* do áudio, ou seja, se o áudio for *sampled* a 48 *kHz*, por cada segundo que o relógio da rede avança, este é incrementado 48000 vezes.

2. **Transporte**
   O transporte descreve como a data, depois de ser codificada e empacotada, é transportada pela rede.

   - ***Network layer***
     Os pacotes de media devem de ser transportados usando o *IP version 4 (IPv4)* [25], de maneira a garantir que as mensagens *multicast* são recebidas e que todas as mensagens indesejadas são filtradas, todos os dispositivos devem de suportar o *IGMPv2*. O *IGMP* deve ser utilizado pelos dispositivos, para requisitar a receção de qualquer mensagem

multicast necessária. Isto inclui as mensagens de sincronização do *PTP*, transmissão de *media* e também mensagens de outros protocolos que possam estar a ser utilizados pelo dispositivo.

- **Qualidade de serviço** (*QoS*)

  Numa rede partilhada não controlada, sendo a *media time-critical*, é necessário existir uma gestão da priorização do tráfego na rede, conhecida como *QoS*. De maneira a facilitar esta gestão, os dispositivos devem implementar o método de *DiffServ* descrito em *RFC 2474* [26]. O *DiffServ* utiliza o campo de *DSCP* no *header* de cada pacote de *IP* para marcar os pacotes de acordo com a sua classe de tráfego. Desta forma a rede consegue identificar e dar prioridade a pacotes que necessitam ser tratados com a maior brevidade possível.

- **Camada de transporte**
  A camada de transporte fornece uma comunicação *end-to-end* entre os vários nós da rede. Os dispositivos devem de utilizar o *RTP*[27] que deve ser transportado por *UDP*[28].

3. **Codificação e Transmissão**
   A codificação descreve como o áudio é digitalizado e partido numa sequência de pacotes que constituem a transmissão. Os seguintes formatos de codificação são suportados:
   L16: *16-bit linear*[27]
   L24: *24-bit linear*[29]
   Todos os dispositivos devem suportar um *sampling rate* de 48 kHz com codificações de *L16* e *L24*. É recomendado que um nó transmissor seja capaz de carregar cada pacote com 48 amostras de áudio a cada milissegundo.

4. **Packet time**

   O *packet time* representa a duração da *media* que está contida em cada pacote. Através do *sampling rate* e do *packet time*, o número de amostras por pacote pode ser calculado.

   Quanto menor for o valor do *packet time* menor será a latência, mas a sobrecarga introduzida com o aumento do número de pacotes pode trazer problemas de processamento aos recetores. Valores altos de *packet time* aumentam a latência e obrigam a que o recetor tenha de fazer *buffering*, o que pode trazer alguns problemas se houver limitações a níveis de memória de quem recebe os pacotes.

   O valor do *packet time* é definido pelo transmissor no descritor de sessão. De forma a garantir a interoperabilidade é requisito que 1 milissegundo de *packet time* seja suportado.

5. **Informação de sessão**

   Um descritor de sessão (*SDP*)[30] é utilizado para especificar informação crítica acerca de cada fluxo de dados, incluindo o formato do codificação, *network addressing* e informação da origem. Por questões de interoperabilidade deve o *SDP* cumprir alguns requisitos e recomendações adicionais.

   - ***Packet time***

     Packet time é descrito pelo *SDP* [30] em dois atributos:

     a=ptime:<milliseconds>[.<milliseconds decila>]
     maxptime=ptime:<milliseconds>[.<milliseconds decila>]

     Multiplicando o *Packet time* pela frequência de *sampling* e arredondado para o inteiro mais próximo, dá-nos o número de *samples* por cada pacote. Os descritores devem de incluir o atributo ***ptime***, indicando o *packet time* desejado. Se mais do que um *packet time* for suportado, o atributo ***maxptime*** indica o *packet time* máximo permitido.

   - ***Clock Source***

     O atributo ***ts-refclk*** específica a referência do relógio de rede usado pela transmissão.

     a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:0

     Neste exemplo o *GMID* do relógio é : ***39-A7-94-FF-FE-07-CB-D0*** e o domínio é 0. Os recetores da transmissão só se devem conectar a transmissores se estiverem a utilizar uma referência de relógio com o mesmo *GMID*.

   - ***RTP* e relógio media**

     A relação entre o relógio do *RTP* e o relógio de media deve ser descrita pelo atributo ***a=mediaclk:direct=<offset>***. A especificação do *offset* deve ser incluída no descritor, este valor indica o desvio que a *timestamp RTP* tem em relação ao relógio de media.

   - **Tipos de *Payload***

     O atributo *rtpmap* é utilizado para indicar o tipo de *payload* presente na transmissão. Uma vez que o valor do tipo de *payload* é dinâmico nos pacotes *RTP*, o recetor não deve assumir que um determinado valor do tipo de *payload* está sempre associado ao mesmo tipo de *payload*. O atributo *rtmap* descreve então o tipo de *payload* associado à variável "tipo de *payload*"no pacote *RTP*. Esta variável não é estática, logo o recetor nunca deve assumir uma relação fixa entre o *payload* que recebe e o tipo de *payload* do pacote.

- **Exemplo descritor de sessão**

  A figura 5.4 exemplifica o formato de um *SDP* de uma *stream* com 8 canais de 24-*bit*, 48*kHz* áudio com 1 milissegundo de *packet time*.

```
v=0
o=- 1311738121 1311738121 IN IP4 192.168.1.1
s=Stage left I/O
c=IN IP4 239.0.0.1/32
t=0 0
m=audio 5004 RTP/AVP 96
i=Channels 1-8
a=rtpmap:96 L24/48000/8
a=recvonly
a=ptime:1
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:domain-nmbr=0
a=mediaclk:direct=963214424
```

Figura 5.4: *Session Descriptor*

5.4 Formato *RTP Payload* para áudio sem compressão

RFC 1551 [27] específica um perfil para o transporte de áudio sem compressão através do protocolo o *RTP*. Neste perfil os valores compreendidos entre o intervalo 96 a 127 podem ser utilizados no cabeçalho do pacote *RTP*, no atributo *payload type*, de forma a que este atributo seja dinâmico e possam ser defendidos novos tipos de carga no pacote.

Também é definido como deve ser transportado áudio que contenha vários canais. Sendo que as amostras pertencentes a todos os canais num determinado instante, devem estar dentro do mesmo pacote *RTP*.

O transmissor pode deferir quantas *frames* de áudio pretende combinar em cada pacote *RTP*, desde que as frames tenham todas o mesmo tamanho. Estas *frames* devem vir ordenadas dentro do pacote *RTP*, sendo que a *frame* mais antiga deve vir imediatamente a seguir ao cabeçalho do pacote.

A *timestamp* do pacote *RTP* reflete o instante em que a primeira *frame* do áudio que vem no pacote foi gerada.

Capítulo 6

Virtualização de um misturador de Áudio

6.1 Requisitos

Na definição de uma solução para virtualização de um misturador de áudio é necessário garantir alguns requisitos:

- A aplicação deve ser capaz de receber áudio de várias fontes.
- De forma a manter a qualidade, a largura de banda fornecida deve ser capaz de suportar transporte de áudio sem compressão na rede.
- O *Iternet Group Management Protocol (IGMP)* deve estar ativo de forma a permitir comunicação por *multicast*.
- A aplicação deve ser usada em cenários em tempo real.
- A aplicação deve ser modular e escalável.
- A aplicação deve permitir a um utilizador selecionar qual dos canais disponíveis à entrada este pretende para a saída.
- A aplicação deve fornecer ao utilizador a informação de quantos canais de áudio estão disponíveis para a operação de comutação.

6.2 Solução Proposta

De forma a criar uma solução para o transporte e manipulação de áudio numa produção televisiva em direto é proposta uma aplicação distribuída na rede que suporta o transporte de vários canais de áudio na camada de *IP*. Esta aplicação, permite a um operador, através do *Web browser*,

realizar operações, como por exemplo, de comutação de áudio, ou seja, entre os várias canais de entrada, selecionar o canal que pretende para a saída da aplicação (módulo *output*) (Figura 6.1).

Com uma aplicação distribuída é possível ter um ambiente flexível e adaptável. Isto é, a simples ação de comutação, com a virtualização da media, passa a ser realizada por *software*, baixando os custos de uma produção e facilitando a escalabilidade e da estrutura.

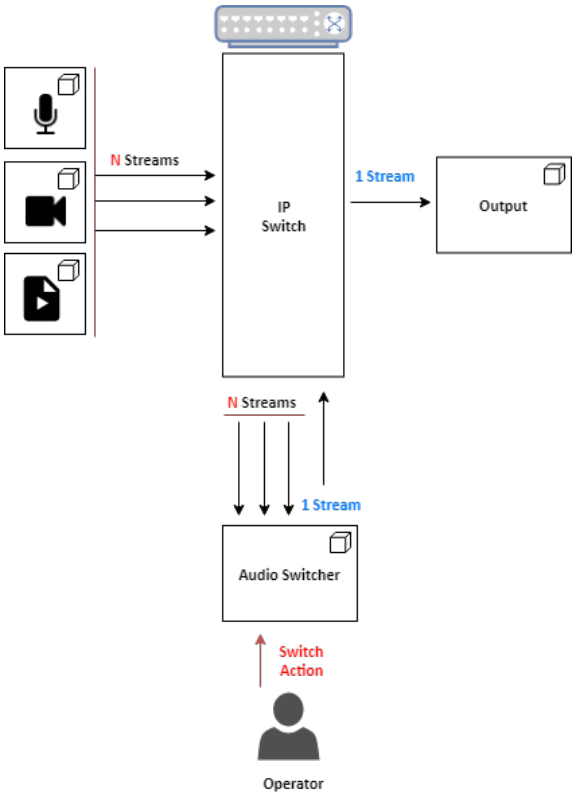

Figura 6.1: Vista *black-box* da aplicação

A aplicação deve ser capaz de ter vários tipos de entrada de áudio, sendo estes *SDI*, transmissões de dados *RTP* ou ficheiros de áudio/vídeo. Devem ser, também, garantidas as questões de sincronização de forma a que todos os fluxos de áudio durante a sua transmissão estejam todos na mesma instância e mantenham coerência em todo o processo até chegarem à fase de saída da aplicação.

6.2.1 Arquitetura

A solução proposta de aplicação é composta por vários módulos (Figura 6.2), todos com funções distintas e específicas. Estes módulos são considerados nós na rede e têm a possibilidade de serem instanciados mais do que uma vez, dependendo do caso de uso. Como todos os nós têm um grande nível de abstração com a rede e a comunicação entre eles é idêntica, a introdução de novos

nós com funcionalidades diferentes acaba por estar facilitada. Nesta arquitetura, todos os módulos pertencem à mesma rede privada.

Uma vez que o projeto onde esta dissertação se insere também inclui o transporte de vídeo, todos os módulos desenvolvidos tiveram isso em conta, de forma a que o a arquitetura proposta seja de fácil integração e, ao mesmo tempo, compatível com o projeto real.

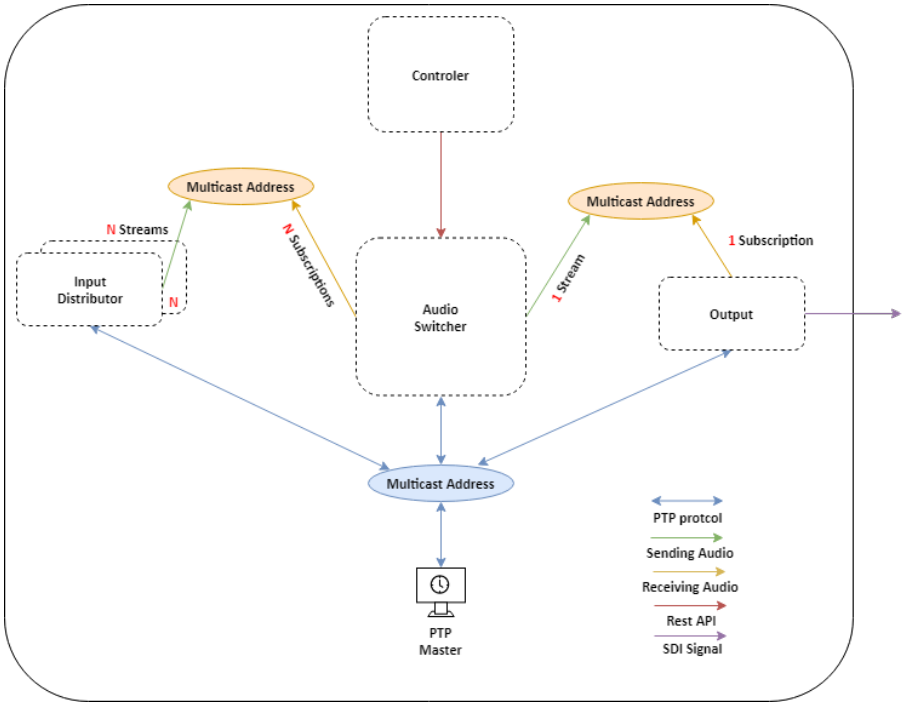

Figura 6.2: Arquitetura proposta

De forma a garantir a qualidade esperada num ambiente de produção, o transporte de áudio entre os módulos é sempre feito sem compressões.

6.2.1.1 *Input Distributor*

O propósito deste módulo é fornecer uma essência sem compressão num endereço *multicast* específico, seguindo as normas do *NMOS*.

Assim, é garantida a interoperabilidade na transferência de essências entre os diferentes módulos.

O *Input Distributor* tem três variantes (figura 6.3). Cada uma das variantes instanciadas deste módulo é responsável por receber o sinal de um *SDI*, um ficheiro de áudio/vídeo ou de um fluxo *MPEG-TS*.

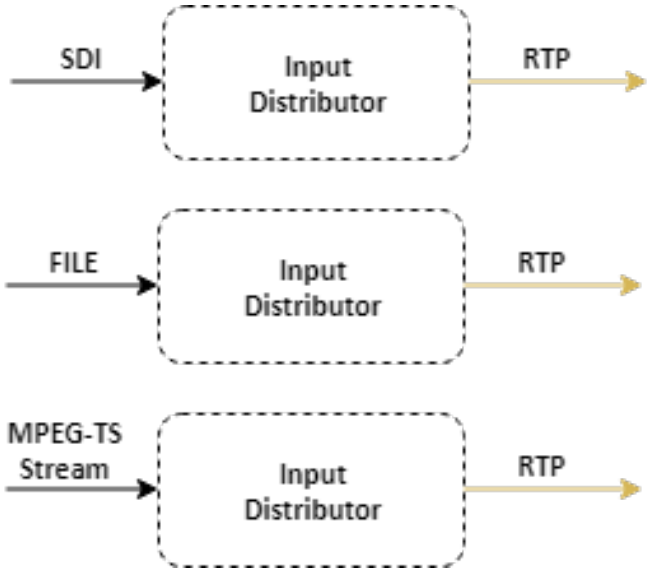

Figura 6.3: *Input Distributor*

6.2.1.2 *Audio Switcher*

O propósito deste módulo (figura 6.4) é selecionar uma entrada específica baseada num evento acionado por um operador.

Este pode receber de várias fontes, áudio sem compressão em fluxos de pacotes *RTP*.

Figura 6.4: *Audio Switcher*

6.2.1.3 *Output*

O *Output* é o módulo da arquitetura responsável por fornecer os resultados das operações de comutações feitas no *Audio Switcher* ao cliente final. Esta componente recebe um fluxo de áudio sem compressão em pacotes *RTP* segundo as especificações do *NMOS* e pode ter como saída um sinal *SDI* sem compressões ou um fluxo *MPEG-TS* que representa áudio comprimido.

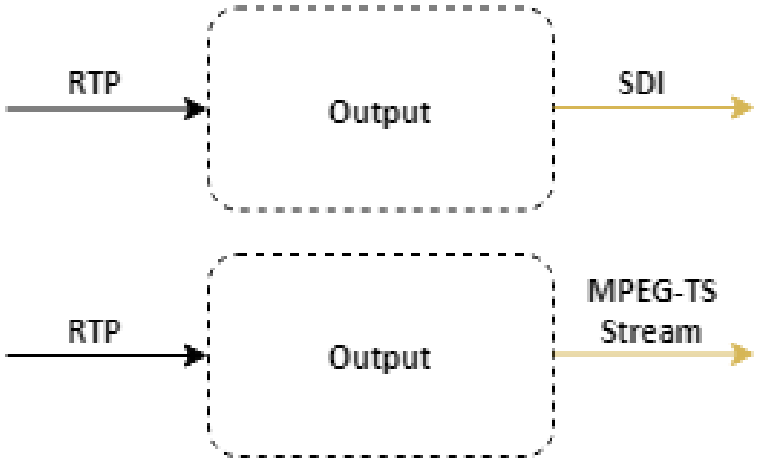

Figura 6.5: *Output*

6.2.2 *Timestamp PTP*

De forma a manter o sincronismo entre todas os fluxos de áudio e, futuramente, entre vídeo e áudio, todos os módulos que fazem transporte de áudio devem ter um relógio sincronizado com o mesmo *Grandmaster* através do *Precision Time Protocol*. Desta forma, utilizando o relógio sincronizado introduzem *timestamps* nos pacotes de *RTP* quando estes são gerados.

6.2.3 Modularidade e Escalabilidade

Todos os módulos são independentes uns dos outros, pelo que, com este nível de abstração, é possível desenvolver e integrar , com alguma facilidade, novos módulos de forma a abranger novas funcionalidades, possibilitando assim a introdução de novos casos de uso. Pode, por exemplo, ser introduzido um módulo novo que armazene o fluxo de áudio resultante da aplicação, ou, então, um módulo que gere uma versão comprimida dos vários fluxos de áudio sem compressão para que seja possível ter uma pré-visualização de cada fluxo no *Controler*. Estes módulos são representados na Figura 6.6 e têm o nome de *Proxy Transcoder* e *Audio Storage*.

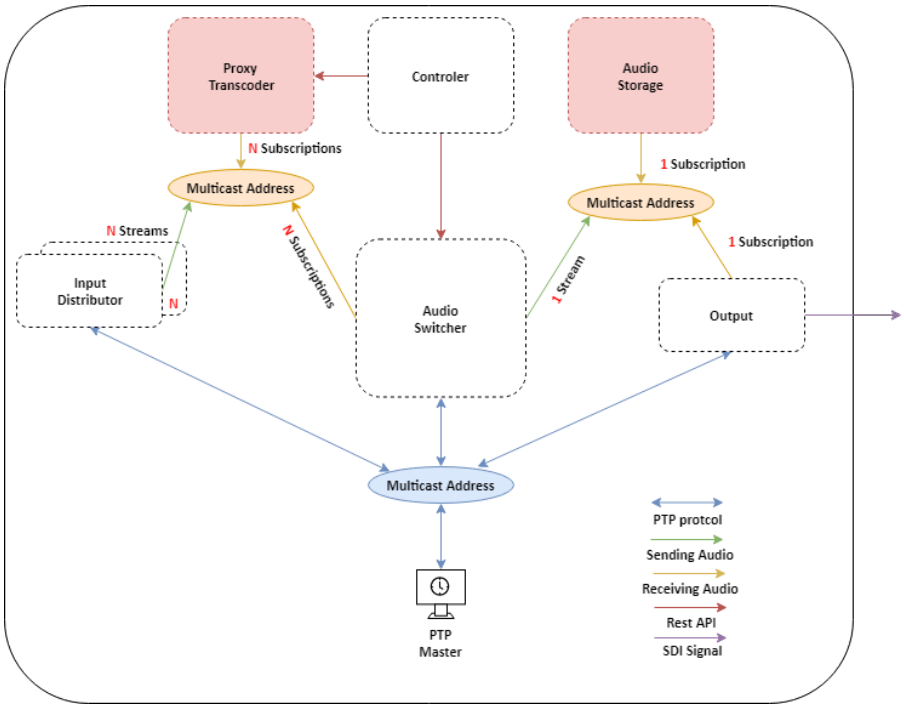

Figura 6.6: Exemplo modularidade

A aplicação é também capaz de escalar, sendo possível lançar várias instâncias de cada módulo de acordo com as necessidades de cada caso de uso. Tendo em conta o *workflow* do projeto em que esta dissertação se insere, o protótipo foi otimizado de forma a suportar várias entradas, um *Audio Switcher* e uma saída (*Output*). Todavia, é possível lançar vários módulos *Audio Switcher*, como se pode ver na figura 6.7. Neste caso existem duas entradas em que apenas uma é selecionada pelo primeiro *Audio Switcher*. No segundo *Audio Switcher* as entradas são um módulo de *Input Distributor* e a saída do primeiro *Audio Switcher*.

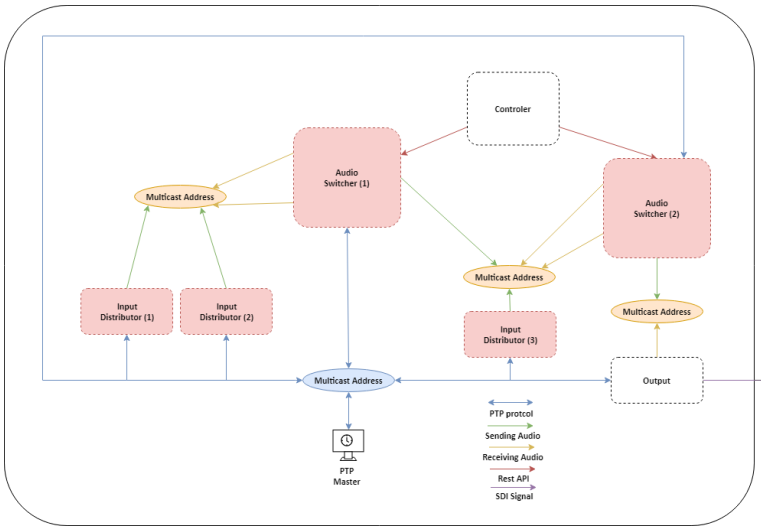

Figura 6.7: Exemplo escalabilidade

6.2.3.1 Novo Caso de Uso

Esta aplicação também pode ser facilmente adaptável para o caso em que estejam a correr várias instâncias da aplicação em diferentes locais, não sendo possível, no entanto, serem lançadas na mesma rede privada. É necessário que o relógio *Grandmaster* de cada aplicação esteja sincronizado com o relógio do mesmo *GPS*, garantindo assim que existe sincronismo com alguma precisão nas aplicações a correr em diferentes locais. Neste caso, é necessário que a saída das diferentes aplicações seja um fluxo *MPEG-TS*, uma vez que não é viável transportar essências sem compressão através da Internet. Este caso de uso é representado pela figura 6.8.

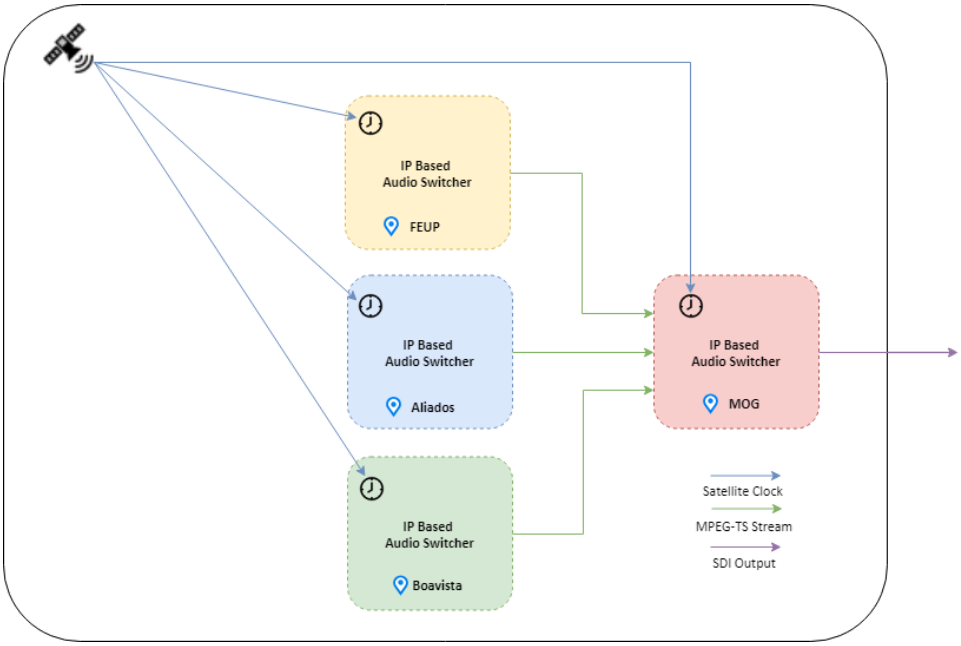

Figura 6.8: Exemplo nós em diferentes localidades

6.3 Protótipo

Foi implementado um protótipo de uma aplicação de forma a validar a arquitetura descrita em 6.2.1.

Neste protótipo foram desenvolvidos os módulos *Input Distributor*, *Audio Switcher* e *Output*, sendo estes capazes de suportar múltiplas instâncias com algumas limitações. O transporte de áudio entre os módulos é todo feito sem compressão e através do protocolo *RTP*. Os módulos são também capazes de gerar e ler ficheiros *SDP* com *metadata* que contém informação acerca dos fluxos de áudio.

Foi também desenvolvido um cliente *PTP* de forma a ser integrado em cada módulo, para que estes possam sincronizar os seus relógios internos com um *Grandmaster* existente na rede.

6.3.1 Limitações do Protótipo

Uma vez que o protótipo implementado tem apenas o propósito de servir como uma prova de conceito, algumas restrições foram definidas durante o desenvolvimento:

- ***Input***: O *Input* deve ser capaz de receber um fluxo *MPEG-TS* e um ficheiro de áudio ou um ficheiro de vídeo com áudio embutido como entrada.
- ***Audio Switcher***: O *Audio Switcher* deve ser capaz de receber pelo menos três fluxos de áudio sem compressão por *RTP*.
- ***Output***: Este módulo deve produzir um fluxo único no formato de *MPEG-TS*.
- A aplicação apenas deve correr em ambiente *windows*.
- **Cliente *PTP***: O cliente *PTP* gera um relógio interno apenas capaz de se sincronizar e comunicar com um *Grandmaster*.
- Se por algum motivo um dos módulos desenvolvidos falhar, este deve ser reiniciado pois não foi desenvolvido nenhum processo para recuperação de módulos.

6.3.2 *Media Processing Library*

A *Media Processing Library* (*MPL*) é uma biblioteca proprietária da *MOG Technologies* utilizada internamente pela empresa no desenvolvimento de novos produtos. É maioritariamente utilizada em soluções de *broadcasting* e fornece uma série de recursos e métodos otimizados para o processamento e transmissão de áudio e vídeo. Ao utilizar esta biblioteca, vários métodos já implementados são aproveitados, facilitando a implementação e sendo, também assegurada a compatibilidade deste projeto com outros produtos da *MOG*.

6.3.3 Implementação

Usando a arquitetura definida na secção 6.2.1, o protótipo desenvolvido focou-se na validação da mesma. Todos os módulos descritos foram implementados (*Input Distributor*, *Audio Switcher*, *Output* e cliente *PTP*).

Devido à extensão da solução proposta, não foi possível integrar o cliente PTP desenvolvido na biblioteca *MPL*, pelo que o relógio que gera as *timestamps* para serem inseridas nos pacotes *RTP* tem um valor inicial fixo.

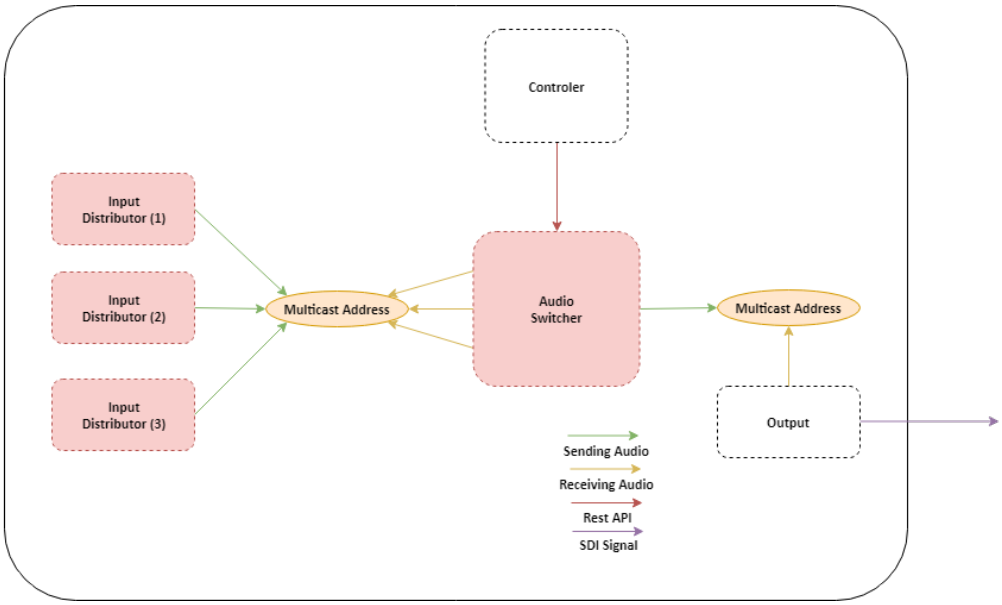

Figura 6.9: Protótipo desenvolvido

Ao desenvolver este protótipo é possível testar:

- Enviar e receber áudio de alta definição sem compressão na camada de *IP*.
- Analisar a capacidade na comutação dos diversos fluxos de áudio em termos de atrasos.
- Validar as especificações propostas pelo *JT-NM NMOS*.
- Obter métricas como largura de banda, recursos gastos em processamento e RAM utilizada.

6.3.3.1 Input Distributor

Foram desenvolvidas três versões deste módulo capazes de suportar entradas de diferentes tipos:

- ***SDI* para *IP***

  A versão do módulo *Input Distributor* que suporta como entrada um sinal *SDI*, é composta por seis componentes principais: *SDI Processor*, *Shared Memory Unit*, *Shared Memory Processor*, *Demuxer*, *Audio RTP Packer* e *Audio RTP Transmissor* (Figura 6.10).

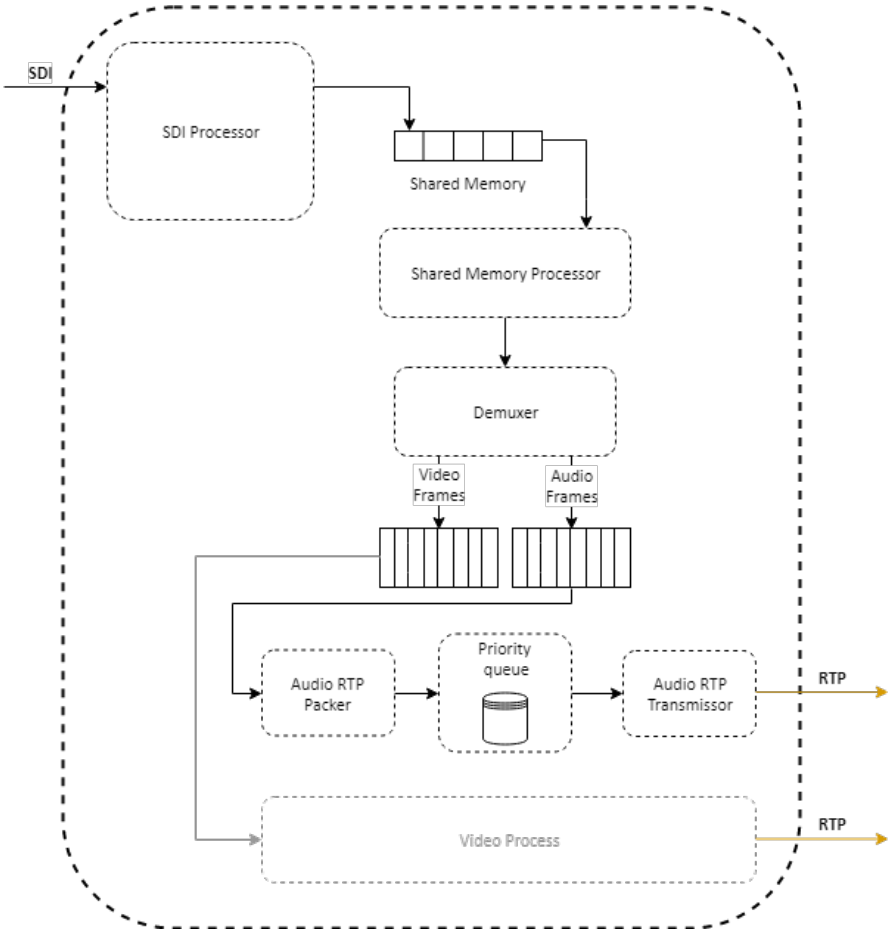

Figura 6.10: Estrutura *Input SDI*

O *SDI Processor* é um processo independente que utiliza um *hardware* especifico, que transforma o sinal *SDI* num sinal digital, escrevendo-o numa memória partilhada. Ao mesmo tempo que vai recolhendo a *data* da memória partilhada, o *Shared Memory Processor* envia-a para o *Demuxer* para que o sinal seja separado em áudio/vídeo *grains* e *metadata*.

- **Ficheiro para *IP***

No cenário em que a entrada é um ficheiro de media, o módulo *Input Distributor* é composto por três componentes principais: *Demuxer*, AUDIO RTP PACKER e *Audio RTP Transmissor* (Figura 6.11).

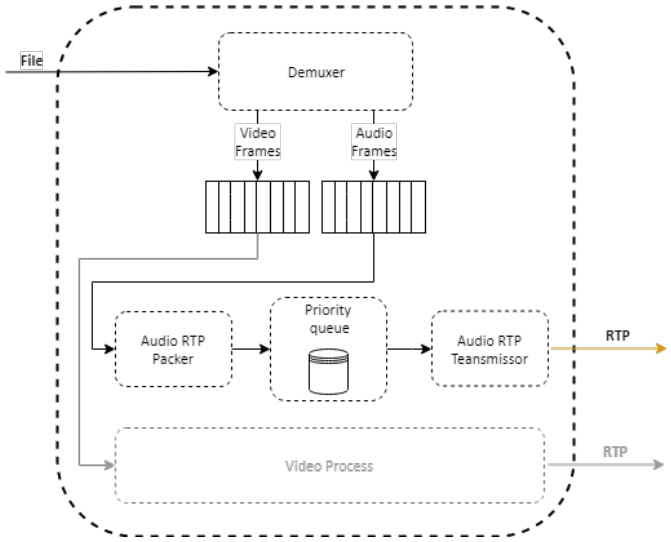

Figura 6.11: Estrutura *Input File*

O *Demuxer* separa o ficheiro em áudio/vídeo *grains* e *metadata*.

- **Fluxo *MPEG-TS***

  Neste caso o módulo é composto por sete componentes principais: *RTP Receiver*, *RTP Unpacker*, *TS Demuxer*, *Grain Factory*, *RTP Packer* e *RTP Transmissor* (Figura 6.12).

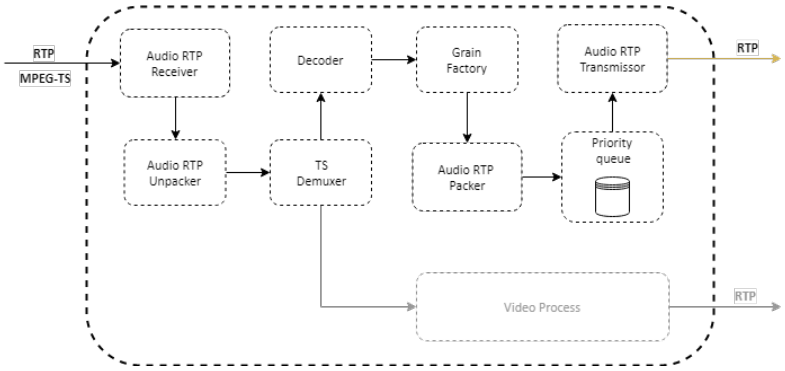

Figura 6.12: Estrutura entrada fluxo *MPEG-TS*

O módulo tem como entrada um fluxo *MPEG-TS*. O *RTP Receiver* é responsável por criar uma sessão *RTP* e receber os pacotes do fluxo *RTP*. Os pacotes são validados pelo *RTP Unpacker* e enviados para o *TS Demuxer*.

Uma vez que o fluxo *MPEG-TS* pode transportar áudio e vídeo, o *TS Demuxer* é utilizado para fazer a separação dos *grains* das diferentes essências.

Depois da separação das essências, o *Decoder* converte a informação que vem comprimida do fluxo *MPEG-TS* para um formato sem compressões e armazena os *grains* na *Grain Factory*.

Após a ocorrência dos processos de *Demuxer* ou *Decode*, dependendo da verão do módulo, é gerada uma *timestamp*. Uma vez que o cliente *PTP* não foi integrado, em vez desta *timestamp* ser gerada tendo como base o seu relógio sincronizado com o *Grandmaster*, esta tem um valor inicial de 0 e é incrementada por 1/*sampling-rate*, neste caso, pode ser 1/48 ou 1/16.

Os processos *Audio RTP Packer* e *Audio RTP Transmissor* são semelhantes em todas as versões do módulo de *Input Distributor*.

O *Audio RTP Packer* é responsável por recolher os *grains* da *Piority queue*, que é preenchida pelo *Demuxer* ou *Decoder*, e inseri-los nos *payloads* dos pacotes *RTP*. Durante este processo é também inserida a *timestamp* gerada nos processos de *Demuxer* ou *Decode* no cabeçalho dos pacotes *RTP*.

Por fim o *Audio RTP Transmissor* envia os pacotes para um endereço de *multicast* especifico dentro de pacotes *UDP*. Este envio é feito a um *rate* constante equivalente ao *sampling-rate*.

Por fim, este módulo gera um ficheiro *SDP* com a *metadata* relativa ao fluxo de áudio que transmite. Em 6.1 podemos ver o *SDP* resultante do módulo *Input Distributor*.

```
1 v=0
2 o=MOGLiveIP 1 1 IN IP4 10.20.155.10
3 s=AudioInput1
4 c=IN IP4 236.114.125.87
5 t=0 0
6 m=audio 5005 RTP/AVP 98
7 c=IN IP4 236.114.125.87
8 a=rtpmap:98 L16/48000/2
9 a=ptime:0.040000
```

Listing 6.1: *SDP* gerado pelo módulo *Input*

6.3.3.2 *Audio Switcher*

O propósito deste módulo é selecionar um dos canais de entrada específico, baseado num evento acionado por um operador. O *Audio Switcher* é formado por seis componentes principais: *Audio RTP Receiver*, *Audio RTP Unpacker*, *Business Logic*, *Grain Factory*, *Audio RTP Packer* e *Audio RTP Transmissor* (Figura 6.13).

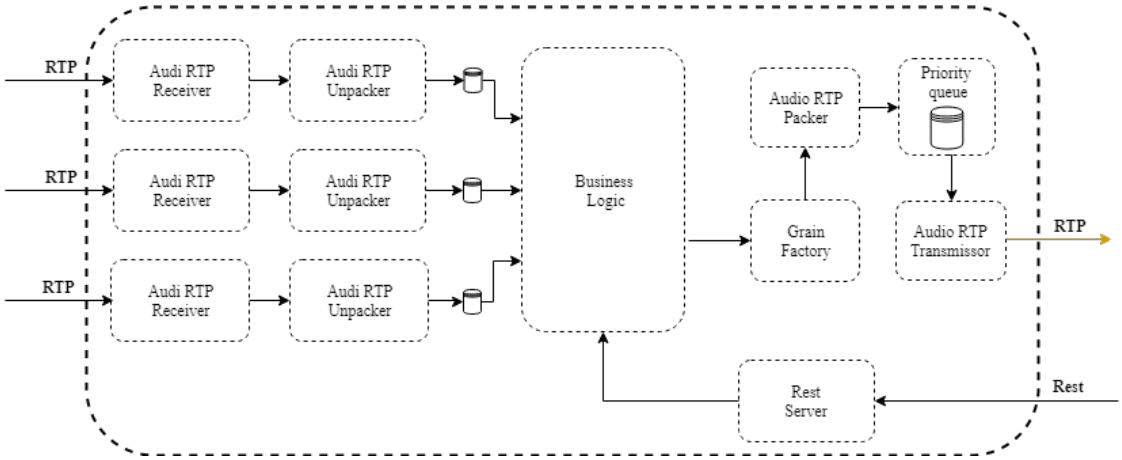

Figura 6.13: Estrutura *Audio Switcher*

O módulo *Audio Switcher* recebe, por *multicast*, áudio não comprimido de vários fluxos *RTP*. Estes são originários de diferentes módulos de *Input Distributor*. O *Audio Switcher* tem acesso aos ficheiros *SDP* gerados pelos módulos *Input Distributor* de forma a ter informação sobre os fluxos que tem de receber.

O processo *Audio RTP Receiver* é responsável por criar uma sessão *RTP* e quando um pacote é totalmente recebido este lança um evento.

Quando um pacote é recebido, o *Audio RTP Unpacker* verifica se o pacote é válido, analisando o cabeçalho do pacote. Depois da validação, o *payload* do pacote é guardado numa *priority queue*. O *Business Logic* recolhe a data de todas as *queues* e envia apenas a data de um canal, previamente selecionado, para o *Grain Factory*, descartando as filas restantes. Este componente é controlado pelo módulo *Controler* e a comunicação é feita através de uma *Rest API*.

Os processos *Audio RTP Packer* e *Audio RTP Transmissor* são semelhantes aos das versões do módulos de *Input Distributor*.

O *Audio RTP Packer* é responsável recolher os *grains* da *Grain Facotry* e inseri-los nos *payloads* dos pacotes *RTP*. Durante este processo é também gerada e inserida uma *timestamp* no *header* dos pacotes.

Por fim o *Audio RTP Transmissor* envia os pacotes para um endereço de *multicast* especifico dentro de pacotes *UDP*. Este envio é feito a um *rate* constante equivalente ao *sampling-rate* da essência do fluxo.

Por fim, este módulo gera um ficheiro *SDP* com a *metadata* relativa ao fluxo de dados de saída.

6.3.3.3 *Output*

- **Versão *SDI***

  O módulo de *Output* para *SDI* é formado por cinco componentes principais: *Audio RTP Receiver*, *Audio RTP Unpacker*, *PCM Parser*, *Multiplexer* e *SDI Producer* (Figura 6.14).

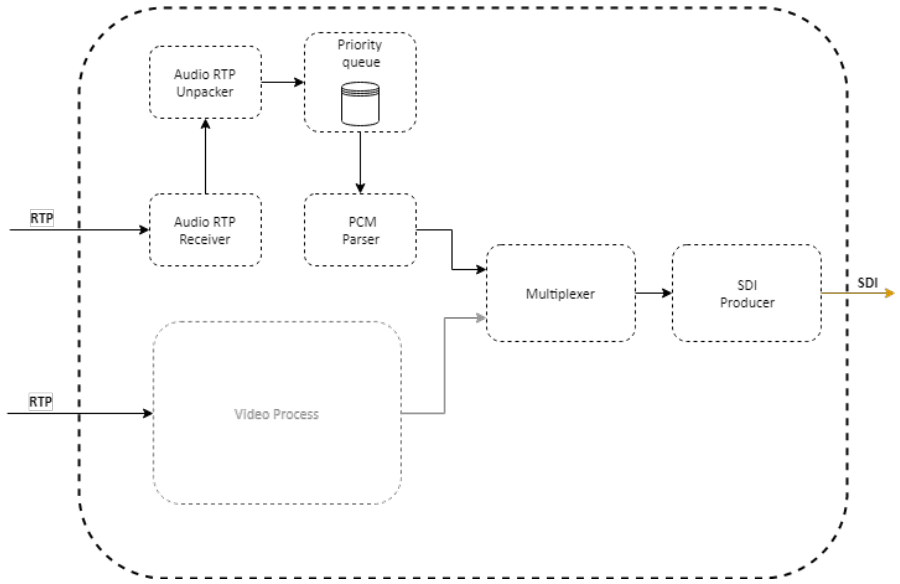

Figura 6.14: *Output SDI*

Este módulo recebe um fluxo de áudio sem compressão.
O *Audio RTP Receiver* é responsável por criar uma sessão *RTP* e lançar um evento quando um pacote é recebido na sua totalidade.
Quando um pacote é recebido, o *Audio RTP Unpacker* analisa o cabeçalho do pacote para verificar se é válido. Depois de validar, o *payload* do pacote, este é armazenado numa *priority queue*.
O *PCM Parser* recolhe a data do áudio sem compressão da *priority queue* e gera um fragmento de áudio. Este é envido para o *Multiplexer*.
Por fim, o *Multiplexer* junta o áudio e vídeo numa saída única, convertendo-a para um sinal *SDI* no processo *SDI Processor*.

- **Versão *MPEG-TS***

  O módulo de *Output* para fluxo *MPEG-TS* é formado por oito componentes principais: *Audio RTP Receiver*, *Audio RTP Unpacker*, *Audio Encoder*, *TS-Multiplexer*, *Grain Factory*, *RTP Packer* e *RTP Transmissor* (Figura 6.15).

  Este módulo recebe um fluxo de áudio sem compressão.
  O *Audio RTP Receiver* e *Audio RTP receiver* são semelhantes à versão do módulo para *SDI*.

  O *Audio Encoder* recolhe a data do áudio sem compressão da *priority queue* e aplica um *PCM encoder* de forma a comprimir o para ser enviado para o *TS-Multiplexer*.

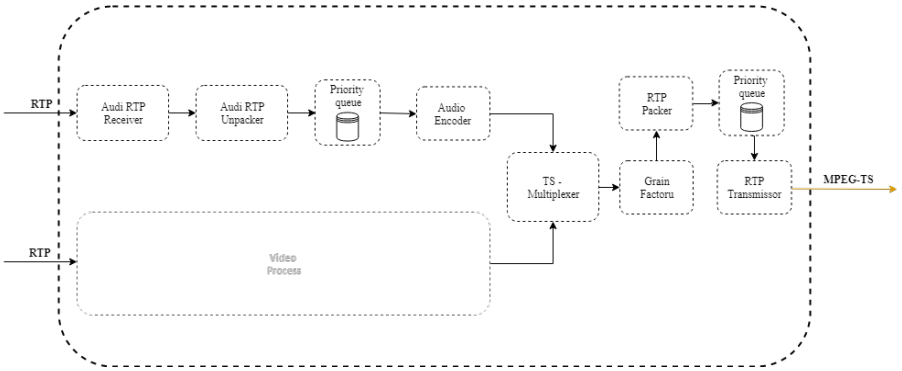

Figura 6.15: Estrutura *Output MPEG-TS*

Por fim, o *TS-Multiplexer* junta o áudio e vídeo comprimidos num *output* único, enviando-o para para fora numa fluxo *RTP* de *MPEG-TS* utilizando os módulos *RTP Packer* e *RTP Transmissor*.

6.3.3.4 *Controler*

O *Controler* consiste numa interface *web* que, de forma simples, permite selecionar qual o fluxo de saída do módulo *Audio Switcer*.

Este módulo comunica com o processo *Business Logic* do módulo *Audio Switcher* através de pedidos efetuados a um servidor *REST*.

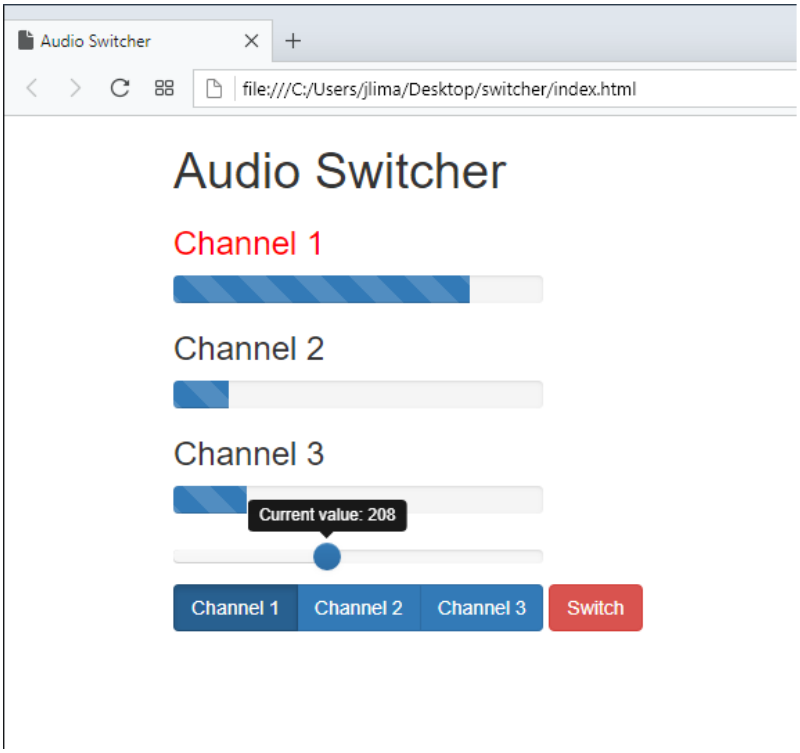

Figura 6.16: Interface *Web*

Esta interface desenvolvida é representada pela Figura 6.16. Neste caso assumiu-se que todas os fluxos de áudio têm a mesma duração e a barra que está abaixo do nome de cada fluxo representa o estado da sua duração. A *scrollbar* lateral permite selecionar o instante, em segundos, que se pretende realizar uma troca, sendo necessário pressionar o botão do canal pretendido e depois o botão de *switch*.

6.3.3.5 Cliente *PTP*

Uma vez que um dos requisitos da aplicação final é correr em ambiente *windows*, não sendo um processo simples alterar o relógio do sistema operativo, o cliente *PTP* implementado será integrado em todos os módulos e terá um relógio próprio.

Foi seguida a sequência de mensagens referida no capítulo 3, secção 3.2.

O cliente desenvolvido é capaz de sincronizar o seu relógio com qualquer *Grandmaster* presente na mesma rede.

Capítulo 7

Resultados e Discussão

Neste capítulo pretende-se analisar os resultados relativos aos testes efetuados, de modo a validar a arquitetura proposta e a sua implementação.

Será apresentado o ambiente de teste, a metodologia utilizada e a discussão dos resultados obtidos.

O cliente *PTP* foi testado independentemente para, posteriormente ser integrado na biblioteca proprietária da *MOG*.

7.1 Metodologia de Teste

7.1.1 Protótipo da Aplicação

Durante a fase de testes, o cenário montado para testar o protótipo da aplicação é representado pela figura 7.1. O ambiente é composto por duas máquinas ligadas na mesma rede privada e as suas especificações são apresentadas na tabela 7.1.

Na primeira máquina foram postas a correr, consoante os testes, uma ou várias instâncias do módulo de *Input Distributor* e, na segunda máquina, uma instância do módulo *Audio Swichet* e outra do módulo *Output*.

Foram realizados quatro testes diferentes com seis ficheiros de áudio diferentes, cujas principais características estão descritas na tabela 7.2:

- **Teste 1:** Foi utilizado apenas um fluxo de entrada com um ficheiro de áudio com *encoding* de 16 bits.
- **Teste 2:** Foi utilizado apenas um fluxo de entrada com um ficheiro de áudio com *encoding* de 24 bits.
- **Teste 3:** Foram utilizados três fluxos de entrada com ficheiros de áudio com *encoding* de 16 bits.
- **Teste 4:** Foram utilizados três fluxos de entrada com ficheiros de áudio com *encoding* de 24 bits.

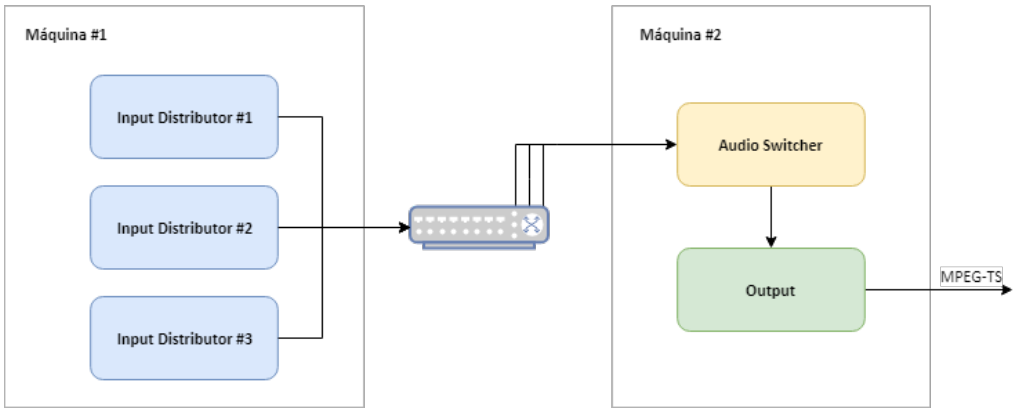

Figura 7.1: Cenário de testes do protótipo

	Máquina #1	**Máquina #2**
CPU	Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.4Ghz 2.39 GHz	Intel(R) Xeon(R) CPU x3470 @ 2.93Ghz 2.93Ghz
RAM	8 GB	8 GB
Placa de Rede	Marvell Yukon 88E8056 PCI-E Gigabit	QLogic BCM5716C Gigabit
Sistema Operativo	Windows 10 Enterprise x64	Windows Server 2016 x64

Tabela 7.1: Máquinas de teste

	Audio File #1	**Audio File #2**	**Audio File #3**	**Audio File #4**	**Audio File #5**	**Audio File #6**
Tamanho do Ficheiro	90.6 MB	90.6 MB	90.6 MB	60.4 MB	60.4 MB	60.4 MB
Extensão do Ficheiro	Wave	Wave	Wave	Wave	Wave	Wave
Duração	5m30s	5m30s	5m30s	5m30s	5m30s	5m30s
Encoding	24 bits	24 bits	24 bits	16 bits	16 bits	16 bits
Sampling rate	48.0 kHz	48.0 kHz	48.0 kHz	48.0 kHz	48.0 kHz	48.0 kHz
Format	PCM	PCM	PCM	PCM	PCM	PCM

Tabela 7.2: Ficheiros áudio de teste

7.1.2 Cliente *PTP*

No cenário de testes para o cliente *PTP* foram utilizadas três máquinas ligadas na mesma rede privada de 10 *gigabit*. Este caso está representado na figura 7.2.

Como *Grandmaster* foi utilizada a solução *PTP4l* numa máquina com o sistema operativo *linux*.

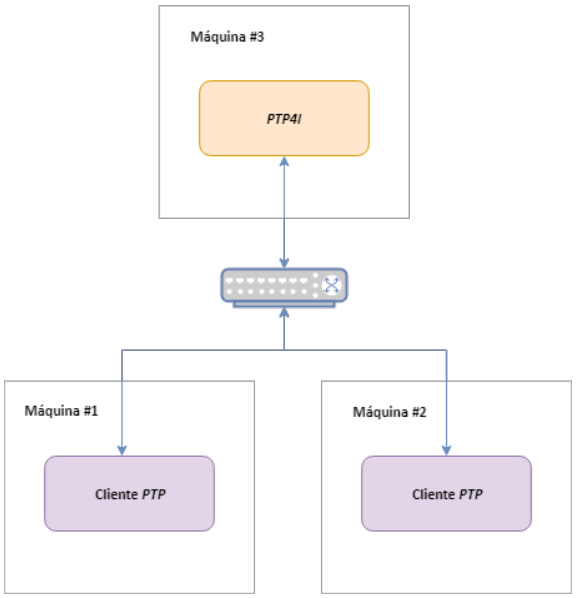

Figura 7.2: Cenário de testes do Cliente *PTP*

7.2 Resultados

De forma a examinar a *performance* do protótipo desenvolvido e do cliente *PTP* foram monitorizadas e analisadas algumas métricas, tais como:

- Uso de *CPU*
- Uso de memória *RAM*
- Largura de banda
- Precisão do relógio *PTP*

7.2.1 Uso de *CPU*

A figura 7.3 apresenta um gráfico com a média do uso de *CPU* para diferentes testes. Como era de esperar, os módulos *Output* e o *Audio Switcher* são os que, em média, usam mais *CPU*. Isto acontece porque o *Output* tem de receber um fluxo sem compressão e gerar o fluxo *MPEG-TS* a

partir da data recebida, enquanto que o *Audio Switcher* tem que processar a entrada de pacotes das vários fluxos de áudio sem compressão e gerar um fluxo de saída.

O cliente *PTP* representa apenas um gasto em média de 0.10% de *CPU*.

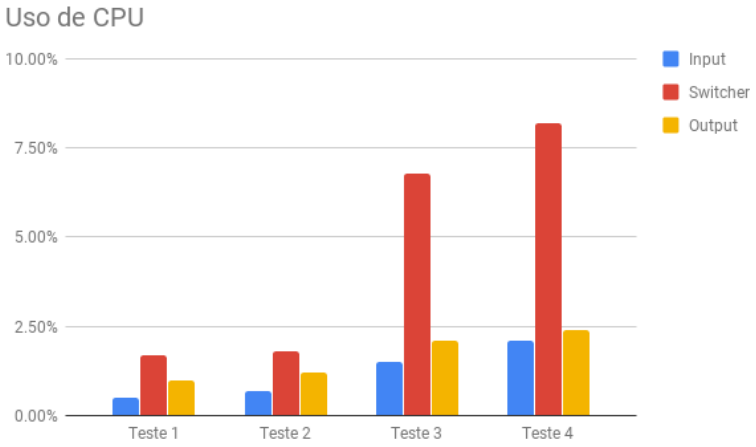

Figura 7.3: Uso de *CPU* em média

7.2.2 Uso de memória *RAM*

A memória *RAM* consumida pelos módulos nos quatro testes efetuados é apresentada na figura 7.4. Os módulos *Audio Switcher* e *Output* são os que consomem mais memória *RAM*. O *Audio Switcher* porque tem que armazenar os pacotes que recebe e os que envia depois de os construir. No caso do *Output* é necessário armazenar os pacotes recebidos e o processo de construção do fluxo *MPEG-TS* também consome alguma memória.

O cliente *PTP* apenas gasta 0.8 *MB* de memória *RAM*.

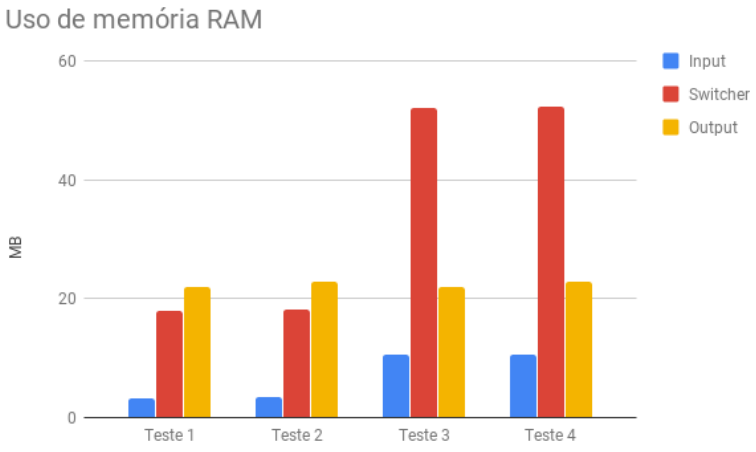

Figura 7.4: Uso de memória *RAM* em média

7.2.3 Largura de banda

No gráfico apresentado na figura 7.5 estão representados os consumos de largura de banda pelos vários módulos nos diferentes testes.

Como esperado, existe um maior consumo nos testes em que são utilizados os ficheiros com codificação de 24 bits. No módulo *Audio Switcher*, existe um maior consumo de entrada do que saída, pois este recebe três fluxos de dados e apenas envia uma para fora.

O cliente *PTP* apenas consome 0.01385 *Mbps*.

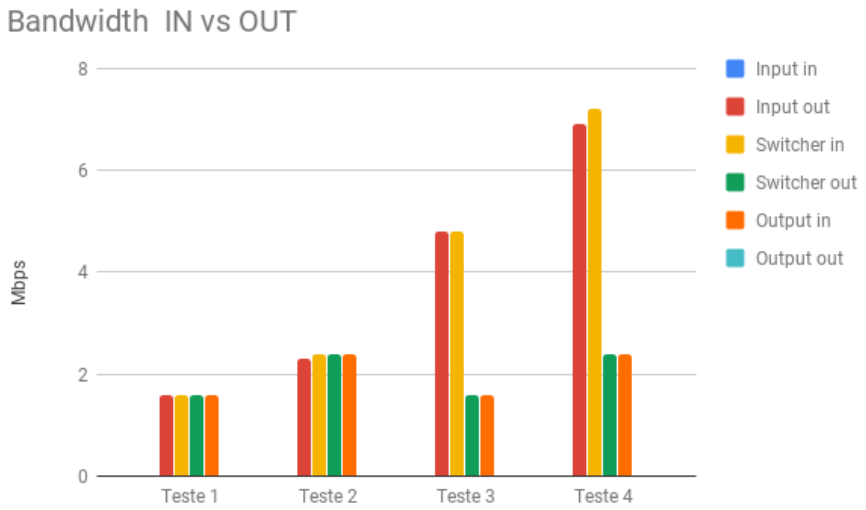

Figura 7.5: Uso de Largura de banda em média

7.2.4 Precisão do relógio *PTP*

Na figura 7.6 é apresentado um gráfico com o erro de relógio do cliente PTP em relação ao relógio do *Grandmaster* presente na rede. Ao longo das 500 iterações de ajustes do relógio, verifica-se que o erro é em média de 200 microssegundos. Uma vez que a tendência é constante (linha a vermelho) e o tempo do *Grandmaster* é incremental, significa que o relógio do cliente *PTP* acompanha esse incremento com alguma precisão, sendo possível mais tarde realizar correções, previstas pelo protocolo, de forma a minimizar este desvio.

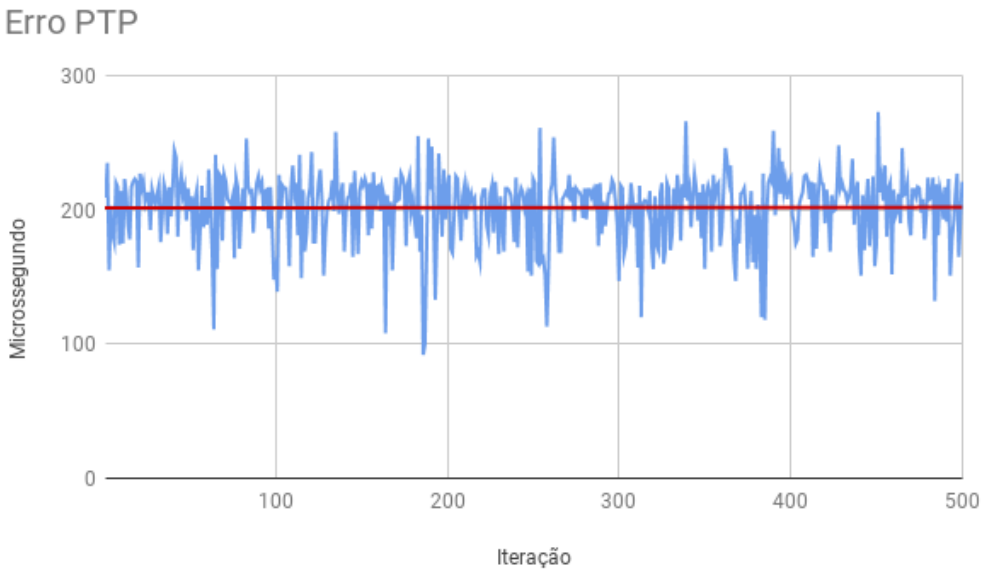

Figura 7.6: Erro PTP

7.2.5 Resultados

Ao analisar as métricas obtidas é possível concluir que os resultados estão dentro dos valores esperados.

No que diz respeito à memória RAM, todos os módulos são capazes de manter um uso muito baixo e constante desta memória.

O gasto de *CPU* em processamento para cada módulo também é relativamente baixo, sendo o *Audio Switcher* o mais crítico devido à quantidade de pacotes que tem de processar.

Relativamente à largura de banda, os valores obtidos numa rede controlada e de 10 gigabit também são ínfimos, sendo obviamente o módulo *Audio Switcher* aquele que apresenta os valores mais altos por ser o que lida com um maior número de fluxos de áudio em certos cenários.

Em relação à sincronização, uma vez que a nível de configurações de rede nada foi feito de forma a garantir o *QoS* e as *timestamps* do *Grandmaster* e cliente *PTP* são geradas por *software*, existe uma maior imprecisão na criação destas *timestamps* e não é garantida a fiabilidade no transporte dos pacotes *PTP* na rede. Como este cliente não é uma versão final, mas sim uma aproximação do protocolo *PTP*, os resultados obtidos são bastante positivos, uma vez que a sincronização é abaixo do milissegundo.

Com os valores obtidos podemos facilmente chegar à conclusão que correr os módulos todos na mesma máquina não apresenta qualquer problema, em termos de utilização de recursos. A tabela 7.3 apresenta esses gastos.

Foram realizados outros testes com o objetivo de obter resultados para verificar e demonstrar que o processo todo de transporte de áudio ocorre sem problemas. Através do *software audacity*[31], foram introduzidos o ficheiro de áudio de entrada no módulo *Input Distributor* e o fi-

CPU	RAM	Largura de Banda
8 %	89 Mb	18.97 Mbps

Tabela 7.3: Recursos utilizados máquina única

cheiro com o fluxo *MPEG-TS* gerado no módulo de *Output* de forma a que estes pudessem ser comparados. Após o *audacity* com o auxilio do *FFmpeg*[32] descomprimir o ficheiro do fluxo *MPEG-TS*, na figura 7.7 é possível verificar que o formato da onda gerada é praticamente equivalente à entrada(canal superior) e à saída(canal inferior), o que leva a concluir que o canal de áudio chega ao destino sem ser danificado.

Com base nestes resultados podemos, também, concluir que o protótipo desenvolvido é compatível com cenários em cascata, ou seja, cenários em que o fluxo de saída gerado por uma instância do protótipo pode servir de entrada de uma outra instância do protótipo.

Por fim, foi utilizado também o *audicity* de modo a poder visualizar a operação de comutação entre canais. Na figura 7.8 é possível observar que ao segundo 55 houve uma troca para um canal que produz um tom constante, observando a onda produzida, e por volta dos 67 segundos houve outra troca para o canal inicial. Podemos, assim, concluir então que a comutação funciona e é feita de forma instantânea.

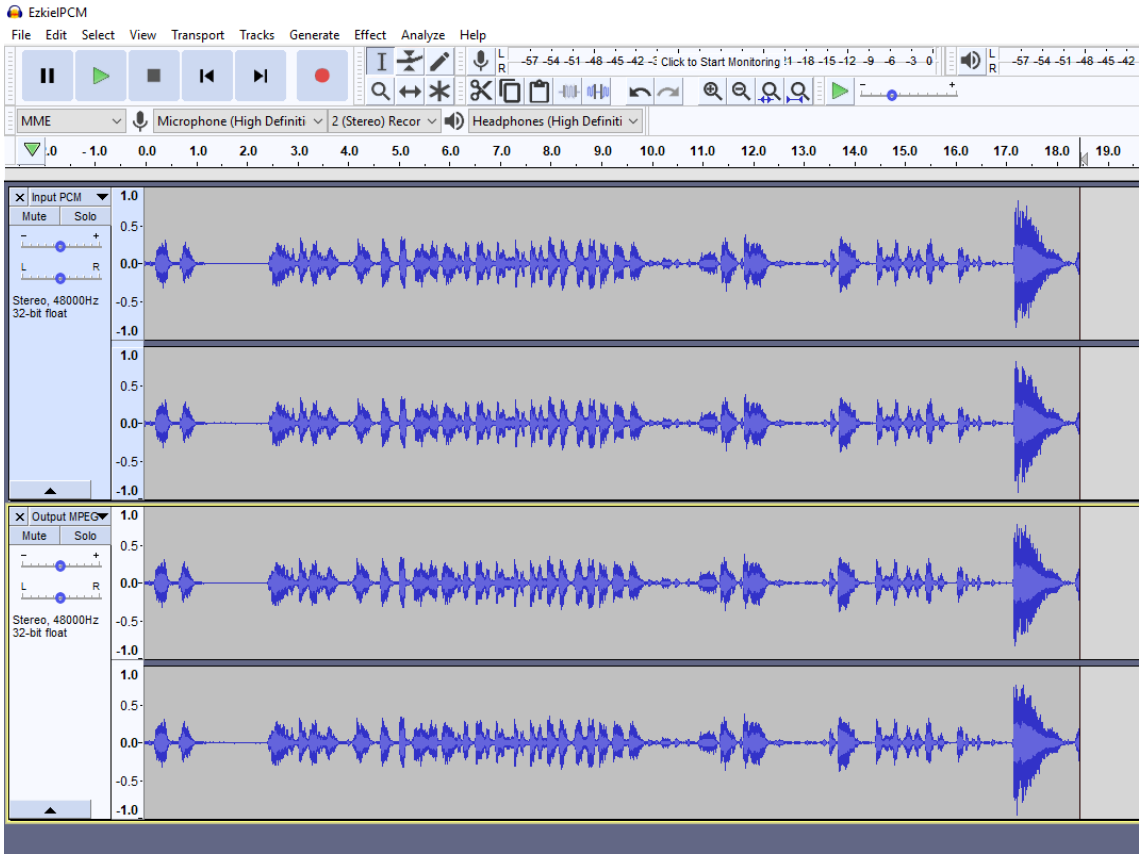

Figura 7.7: Onda *input vs output*

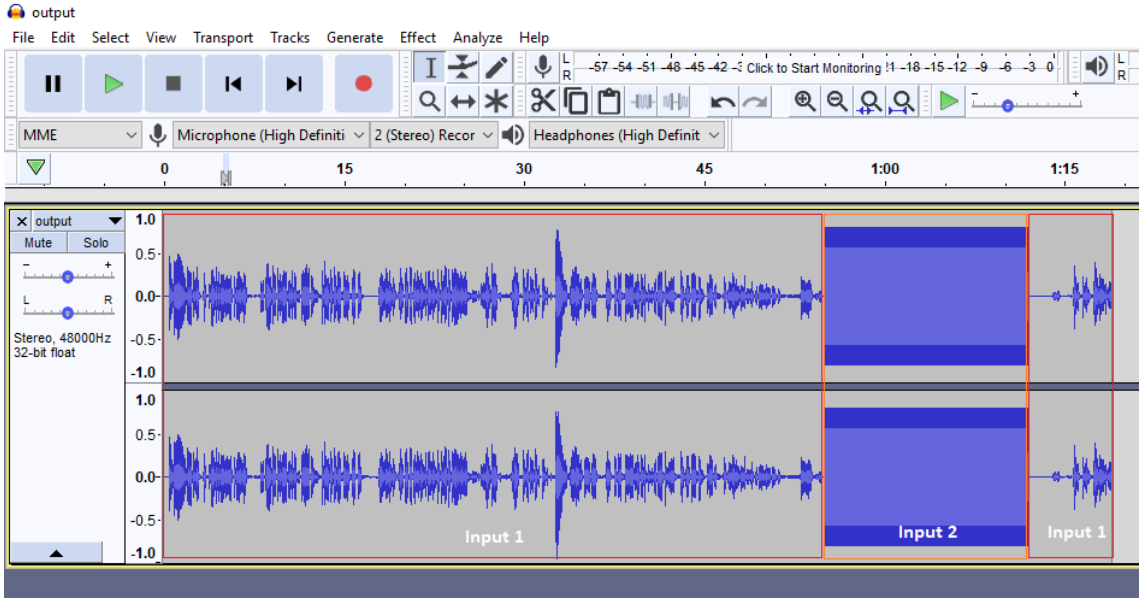

Figura 7.8: Operação de comutação

Capítulo 8

Conclusões e Trabalho Futuro

Nesta dissertação propôs-se uma arquitetura de alto nível e desenvolveu-se um protótipo de uma aplicação capaz de transportar e comutar diferentes canais de áudio numa rede de *IP*, com base no que é proposto pelo *JT-NM* e *NMOS*, com o objetivo de desenvolver uma aplicação distribuída sobre *IP*. Através do protótipo desenvolvido, verificou-se a viabilidade da arquitetura, as possibilidades da sua expansão e a facilidade com que pode ser adaptável a novos casos de uso. Foi também desenvolvido um cliente *PTP* que é capaz de sincronizar o seu relógio interno com um relógio *Grandmaster* presente na mesma rede.

Sendo assim o trabalho desenvolvido nesta dissertação, apresenta-se como o inicio de uma solução possível de ser trabalhada e melhorada no futuro.

8.0.1 Satisfação dos Objetivos

Os objetivos principais inicialmente propostos para esta dissertação foram atingidos. O trabalho realizado permitiu compreender algumas das soluções existentes para ajudar a transição da indústria televisiva para o *IP*. A arquitetura proposta não só é capaz de suportar os objetivos definidos, como pode também, ser facilmente expansível e adaptável a novos casos de uso.

O protótipo desenvolvido testou com sucesso a arquitetura proposta, demonstrando que é possível transportar a infraestrutura atual para *IP*.

8.0.2 Trabalho Futuro

Numa primeira fase, um dos trabalhos a realizar será integrar o cliente *PTP* na biblioteca *MPL*, de forma a que este possa ser utilizado pelos módulos para se sincronizarem com o *Grandmaster*.

Otimizar e completar o cliente *PTP* de forma a melhorar a precisão já obtida.

Completar o processo de geração do ficheiro *SDP* com as informações do relógio.

Um novo módulo pode ser desenvolvido para permitir que cada nó se registe no sistema e descubra os outros nós já presentes, criando, assim, um ambiente onde todos os módulos podem comunicar entre si sem conhecerem previamente as suas localizações.

Tratar das configurações necessários a nível da infraestrutura para garantir o mínimo de atraso possível na circulação de pacotes na rede.

Desenvolver um novo módulo que gere uma versão de baixa resolução dos canais, para que seja possível apresentar na interface *web* de controlo uma forma do utilizador perceber o que está a tocar no momento em cada canal.

Estudar uma solução para a introdução de efeitos na transição entre canais após o processo de comutação ser ativado

Virtualizar os módulos em *containers* de forma a poder tornar a solução *cloud-based* tirando um maior proveito da sua modularidade.

Referências

[1] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, e Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, Julho 2003. URL: https://rfc-editor.org/rfc/rfc3550.txt, doi:10.17487/RFC3550.

[2] Philip J Cianci. *Technology and Workflows for Multiple Channel Content Distribution: Infrastructure implementation strategies for converged production*. Focal Press, 2012.

[3] T. Neagoe, V. Cristea, e L. Banica. Ntp versus ptp in com puter networks clock synchronization. Em *2006 IEEE International Symposium on Industrial Electronics*, volume 1, páginas 317–362, July 2006. doi:10.1109/ISIE.2006.295613.

[4] Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, páginas 1–300, July 2008. doi:10.1109/IEEESTD.2008.4579760.

[5] T. Kojima, J. J. Stone, J. R. Chen, e P. N. Gardiner. A practical approach to ip live production. *SMPTE Motion Imaging Journal*, 124(2):29–40, March 2015. doi:10.5594/j18514.

[6] Joint task force on networked media (jt-nm). http://jt-nm.org/index.shtml. (Accessed on 06/18/2018).

[7] Amwa-tv/nmos: Networked media open specifications. https://github.com/AMWA-TV/nmos. (Accessed on 06/19/2018).

[8] Nestor Amaya. Aes67 for audio productions: Background applications and challenges., 2016. URL: https://www.smpte.org/sites/default/files/users/user27446/AES67%20for%20Audio%20Production-Background%20Applications%20and%20Challenges.pdf.

[9] conclutionsott.pdf. https://www.anacom.pt/streaming/conclutionsOTT.pdf?contentId=1382167&field=ATTACHED_FILE. (Accessed on 03/04/2018).

[10] media-industry-transition-ip-making-technology-case.pdf. https://www.cisco.com/c/dam/en_us/solutions/industries/downloads/media-industry-transition-ip-making-technology-case.pdf. (Accessed on 03/04/2018).

[11] Chic – cooperative holistic view on internet and content. https://chic.mog-technologies.com/. (Accessed on 03/04/2018).

[12] Charles M Kozierok. *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*. No Starch Press, 2005.

[13] Dr. Steve E. Deering. Host extensions for IP multicasting. RFC 1112, Agosto 1989. URL: https://rfc-editor.org/rfc/rfc1112.txt, doi:10.17487/RFC1112.

[14] Jim Martin, Jack Burbank, William Kasch, e Professor David L. Mills. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905, Junho 2010. URL: https://rfc-editor.org/rfc/rfc5905.txt, doi:10.17487/RFC5905.

[15] January 2015 | society of motion picture & television engineers. https://www.smpte.org/publications/past-issues/January-2015. (Accessed on 06/18/2018).

[16] St 2022-6:2012 - smpte standard - transport of high bit rate media signals over ip networks (hbrmt). *SMPTE ST 2022-6:2012*, páginas 1–16, Oct 2012. doi:10.5594/SMPTE.ST2022-6.2012.

[17] St 2110-10:2017 - smpte standard - professional media over managed ip networks: System timing and definitions. *ST 2110-10:2017*, páginas 1–17, Nov 2017. doi:10.5594/SMPTE.ST2110-10.2017.

[18] St 2110-30:2017 - smpte standard - professional media over managed ip networks: Pcm digital audio. *ST 2110-30:2017*, páginas 1–9, Nov 2017. doi:10.5594/SMPTE.ST2110-30.2017.

[19] M. Yonge e B. Harris. AES standard for network and file transfer of audio-Audio-file transfer and exchange-Radio traffic audio delivery extension to the Broadcast Wave file format. *JOURNAL OF THE AUDIO ENGINEERING SOCIETY*.

[20] Ravenna network. URL: https://www.ravenna-network.com/.

[21] Audinate - dante audio networking, av's leading technology. https://www.audinate.com/. (Accessed on 06/03/2018).

[22] Livewire+ aes67 aoip networking. https://www.telosalliance.com/Axia/Livewire-AoIP-Networking. (Accessed on 06/03/2018).

[23] Wheatnet-ip technology overview. http://wheatstone.com/blades-ip-audio-network/wheatnet-ip-technology-overview. (Accessed on 06/03/2018).

[24] Qualified q-lan audio/video switches - q-sys networking & 3rd party telephony integration - resources - q-sys platform - products - systems - qsc. https://www.qsc.com/systems/products/q-sys-platform/resources/q-sys-networking-3rd-party-telephony-integration/qualified-q-lan-audiovideo-switches/. (Accessed on 06/03/2018).

[25] Internet Protocol. RFC 791, Setembro 1981. URL: https://rfc-editor.org/rfc/rfc791.txt, doi:10.17487/RFC0791.

[26] Fred Baker, David L. Black, Dr. Kathleen M. Nichols, e Steven L. Blake. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, Dezembro 1998. URL: https://rfc-editor.org/rfc/rfc2474.txt, doi:10.17487/RFC2474.

[27] Stephen L. Casner e Henning Schulzrinne. RTP Profile for Audio and Video Conferences with Minimal Control. RFC 3551, Julho 2003. URL: https://rfc-editor.org/rfc/rfc3551.txt, doi:10.17487/RFC3551.

[28] User Datagram Protocol. RFC 768, Agosto 1980. URL: https://rfc-editor.org/rfc/rfc768.txt, doi:10.17487/RFC0768.

[29] Carsten Bormann, Stephen L. Casner, Katsushi Kobayashi, e Akimichi Ogawa. RTP Payload Format for 12-bit DAT Audio and 20- and 24-bit Linear Sampled Audio. RFC 3190, Janeiro 2002. URL: https://rfc-editor.org/rfc/rfc3190.txt, doi:10.17487/RFC3190.

[30] Colin Perkins, Mark J. Handley, e Van Jacobson. SDP: Session Description Protocol. RFC 4566, Julho 2006. URL: https://rfc-editor.org/rfc/rfc4566.txt, doi:10.17487/RFC4566.

[31] Audacity ® | free, open source, cross-platform audio software for multi-track recording and editing. https://www.audacityteam.org/. (Accessed on 07/02/2018).

[32] Ffmpeg. https://www.ffmpeg.org/. (Accessed on 07/02/2018).